**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И.Раззакова**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра «Программное Обеспечение Компьютерных Систем»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ

«ОСНОВЫ КОНСТРУИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

Для студентов подготовки бакалавров по кредитной технологии обучения по направления 710400 «Программная инженерия»

Бишкек 2017

**РАССМОТРЕНО ОДОБРЕНО**

**На заседании кафедры Методическим советом ФИТ**

**«Программное обеспечение**

**компьютерных систем»**

**Прот. № 11 от 27.03.2017г. Прот. №11 от 28.03.2017г.**

**УДК 004.415.2**

**Составитель – Хоменко Т.Н., Семененко А.С.**

**Основы конструирования программного обеспечения: методические указания к выполнению лабораторных работ** /**Кырг. гос. техн. ун-т им. И. Раззакова, Бишкек, 2017, 48с.**

По учебному плану для студентов бакалавров по кредитной технологии обучения по направлению 710400 «Программная инженерия» дисциплина «Основы конструирования программного обеспечения» входит в цикл обязательных дисциплин.

Цели этой дисциплины:

* формирование умений разрабатывать алгоритмы решения разнообразных задач, графической и программной реализации алгоритмов;
* формирование систематизированного представления о современном комплексе задач, методов и стандартах программной инженерии.

Методические указания рассматривают основные виды алгоритмов для решения разнообразных задач, графическую форму представления алгоритмов, создание псевдокодов для полуформализованного описания алгоритмов на условном алгоритмическом языке и программную реализацию алгоритмов. Кроме этого, методические указания рассматривают основные понятия, цели и задачами тестирования программного обеспечения и создание тестовых вариантов.

**Библиогр. 15 названий.**

**Рецензент доцент Мусина И.Р.**

**Оглавление**

[**Введение** 4](#_Toc478678087)

[Лабораторная работа №1 5](#_Toc478678088)

[**Построение линейных алгоритмов** 5](#_Toc478678089)

[Лабораторная работа №2 11](#_Toc478678090)

[**Построение разветвляющихся алгоритмов** 11](#_Toc478678091)

[Лабораторная работа №3 16](#_Toc478678092)

[**Организация структуры ветвления посредством** 16](#_Toc478678093)

[**оператора *switch-case.*** 16](#_Toc478678094)

[Лабораторная работа №4 19](#_Toc478678095)

[**Построение циклических алгоритмов** 19](#_Toc478678096)

[Лабораторная работа №5 25](#_Toc478678097)

[**Одномерные массивы** 25](#_Toc478678098)

[Лабораторная работа №6 30](#_Toc478678099)

[**Двумерные массивы** 30](#_Toc478678100)

[Лабораторная работа №7 35](#_Toc478678101)

[**Создание тестовых вариантов** 35](#_Toc478678102)

[**Тестирование «черного ящика».** 36](#_Toc478678103)

[**Тестирование «белого ящика**». 37](#_Toc478678104)

[Лабораторная работа №8 40](#_Toc478678105)

[**Реализации линейных, разветвляющихся и циклических** 40](#_Toc478678106)

[**алгоритмов на VBA.** 40](#_Toc478678107)

[**Литература** 48](#_Toc478678108)

# 

# Введение

В разных источниках можно встретить разнообразные варианты определения понятия *конструирование программного обеспечения*. Наиболее часто встречаемым из них является то, что конструирование программного обеспечения – это совокупность детального проектирования, кодирования и тестирования.

Детальное проектирование – декомпозиция системы до уровня очевидно реализуемых функций, модулей и процедур. На этом этапе программное обеспечение должно быть разбито на множество функций и определено, каким образом и в какой последовательности будет реализована каждая из них (какие входные/выходные данные и т.д.). Только после того, как мы ответили на вопрос, какие функции должны быть реализованы, мы можем приступать непосредственно к написанию кода и тестированию.

Так как дисциплина «Основы конструирования программного обеспечения» изучается студентами на первом курсе, то процесс конструирования рассматривается на примере задач линейных, ветвящихся и циклических алгоритмов. Построение и анализ описаны простым человеческим языком и с применением псевдокода, который понятен любому, кто хоть в небольшой степени знаком с программированием, а пояснения принципов их работы даны без излишней математической строгости и требуют лишь элементарных знаний.

# Лабораторная работа №1

## **Построение линейных алгоритмов**

**Цель работы**

Ознакомление студентов с линейными алгоритмами и построением блок-схем таких алгоритмов.

1. **Теоретические сведения:**

Человек ежедневно встречается с необходимостью следовать тем или иным правилам, выполнять различные инструкции и указания. Обычно любые инструкции и правила представляют собой последовательность действий, которые необходимо выполнить в определенном порядке. Для решения задачи надо знать, что дано, что следует получить и какие действия и в каком порядке следует для этого выполнить. Предписание, определяющее порядок выполнения действий над данными с целью получения искомых результатов, и есть алгоритм.

**Алгоритм** – описание последовательности действий (план), строгое исполнение которых приводит к решению поставленной задачи за конечное число шагов.

***Свойства алгоритмов:***

**Понятность для исполнителя -** исполнитель алгоритма должен понимать, как его выполнять. Иными словами, имея алгоритм и произвольный вариант исходных данных, исполнитель должен знать, как надо действовать для выполнения этого алгоритма**.**

**Дискретность** (от лат. discretus — разделённый, прерывистый, раздельность) - алгоритм должен представлять процесс решения задачи как последовательное выполнение простых шагов (этапов);

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

**Конечность** -за конечное число шагов алгоритм либо должен приводить к решению задачи, либо после конечного числа шагов останавливаться из-за невозможности получить решение с выдачей соответствующего сообщения, либо неограниченно продолжаться в течение времени, отведенного для исполнения алгоритма, с выдачей промежуточных результатов;

**Массовость** - один и тот же алгоритм можно использовать с разными исходными данными.

***Основные виды алгоритмов:***

1. **Линейный алгоритм –** алгоритм, в котором происходит описание действий, выполняющихся однократно в заданном порядке;
2. **Разветвляющийся алгоритм -** алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий;
3. **Циклический алгоритм** - алгоритм, в котором происходит описание действий, которые должны повторятся указанное число раз или пока не выполнено заданное условие.

На практике наиболее распространены следующие **формы представления алгоритмов:**

1. **Словесная -** запись на естественном языке;
2. **Графическая -** изображения из графических символов;
3. **Псевдокоды -** полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.;
4. **Программная -** тексты на языках программирования**.**

Для более наглядного представления алгоритма широко используется графическая форма – **блок-схем**а, которая составляется из стандартных графических объектов.

При графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий. В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т.п.) соответствует геометрическая фигура.

|  |  |
| --- | --- |
| Обозначение блока | Выполняемая функция |
|  | Начало или Конец алгоритма |
|  | Вычисляемые действия (функции) |
|  | Проверка условия: выбор одного из двух направлений |
|  | Ввод или Вывод данных |
|  | Организация циклических процессов |
|  | Направление линий потока - стрелка |

**Рис.1.1**. Наиболее часто употребляемые блоки.

***Стадии создания алгоритма:***

1. Алгоритм должен быть представлен в форме, понятной человеку, который его разрабатывает (определить цель, наметить план действий).

2. Алгоритм должен быть представлен в форме, понятной тому объекту (в том числе и человеку), который будет выполнять описанные в алгоритме действия (выбрать среду и объект алгоритма, детализировать алгоритм).

Объект, который будет выполнять алгоритм, обычно называют *исполнителем.* Идеальными исполнителями являются машины, роботы, компьютеры. Алгоритм, записанный на «понятном» компьютеру языке программирования, называется *программой.*

***Правила построения алгоритмов на языке блок-схем:***

1. Блок-схема строится сверху вниз;
2. В любой блок-схеме имеется один элемент, соответствующий началу и один элемент, соответствующий концу;
3. Должен быть хотя бы один путь из начала блок-схемы к любому элементу;
4. Должен быть хотя бы один путь от каждого элемента в коней блок-схемы.

**Линейный алгоритм**

Линейный алгоритм – описание действий, которые выполняются однократно в заданном порядке. Исполнитель выполняет действия последовательно, одно за другим в том порядке в котором они следуют.



**Рис.1.2**. Блок-схема линейного алгоритма.

Для того, чтобы задача стала максимально понятной, нужно составить постановку задачи.

**Постановка задачи** — точная формулировка условий **задачи** с описанием входной и выходной информации. *Входная информация по задаче* — данные, поступающие на вход задачи и используемые для её решения. *Выходная информации* – ожидаемое решение задачи.

*Общий вид постановки задачи:*

Входные переменные:

[Тип данных переменной] [Имя переменной] - [описание переменной]

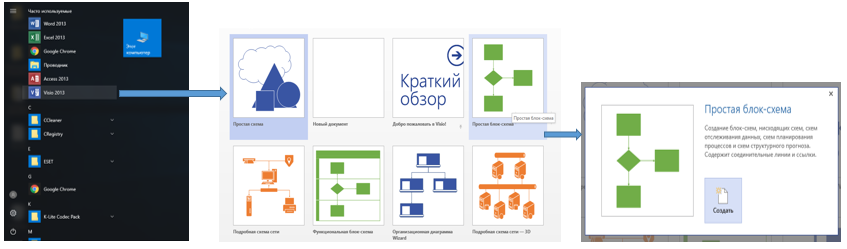
…………………………

Выходные переменные:

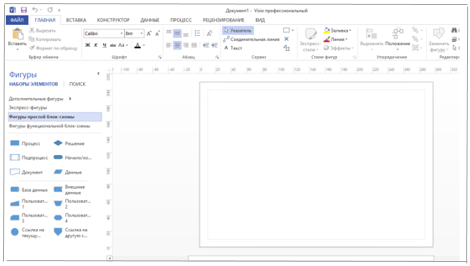
[Тип данных переменной] [Имя переменной] - [описание переменной]

1. **Задания к работе**
2. Составить постановку задачи нахождения суммы двух натуральных чисел А и В.
3. Построить блок-схему алгоритма решения задачи №1.
4. Составить постановку задачи нахождения площади треугольника, если известны 3 его стороны A, B и C.
5. Построить блок-схему алгоритма решения задачи №3.
6. **Порядок выполнения лабораторной работы**

* Откройте *Пуск -> Microsoft Office -> Visio 2013 -> Простая блок схема -> Создать*.



**Рис. 1.3.** Создание нового документа.



**Рис. 1.4.** Рабочая область MS Visio 2013

* Переименуйте *Страница-1* на Сумма 2-х чисел, чтобы в дальнейшем не путаться в задачах, так как в одной рабочей книге задач будет более одной.
* Во вкладке *Главная* выберите пункт *Текст* и разместите его в удобное для Вас место на рабочем листе и приступайте к выполнению заданий.

**Выполнение задания №1.**

Постановка задачи нахождения суммы двух натуральных чисел**:**

*Входные переменные:*

Целое A – первое число (слагаемое);

Целое B – второе число(слагаемое);

*Выходные переменные:*

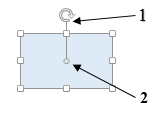
Целое Sum – сумма чисел А и В.

**Выполнение задания №2.**

* Откройте Ф*игуры простой блок-схемы* в наборе элементов.

Если назначение фигуры вам не знакомо, то можно задержать на ней указатель мыши, и появится всплывающая подсказка, которая сообщит, какую команду выполняет эта фигура.

При выделении фигуры вокруг нее появляется прямоугольное поле выделения. Это поле имеет восемь манипуляторов выделения: по одному для каждого из четырех углов и по одному на каждой из сторон поля выделения. Поверх поля выделения появляется один манипулятор вращения (1) и манипулятор вращения в центре фигуры (2).



**Рис 1.5.** Редактирование фигуры.

С помощью манипуляторов можно изменять размеры фигуры и ее угол поворота. Если задержать указатель поверх управляющего манипулятора, то появится всплывающая подсказка, говорящая о том, что можно делать с этим управляющим манипулятором. Для поворота фигуры необходимо перетащить манипулятор вращения. Чтобы пропорционально изменить ее размер, необходимо перетащить один из угловых манипуляторов выделения.

При перемещении фигуры в нужное место отпустите кнопку мыши. Фигура останется выделенной, а строка состояния будет отображать ее ширину, высоту и угол поворота Щелкните мышью на области вставки, и выделение фигуры будет снято.

Каждая следующая фигура добавляются в документ таким же образом.

Чтобы перенести фигуры в другое место, необходимо их выделить. Можно выделить каждую фигуру по отдельности, поместить указатель мыши поверх фигуры, и поочередно перенести. Можно выделить несколько фигур одновременно рамкой так же, как и в других графических приложениях.

* Путем перетаскивания необходимых блоков в рабочую область нарисуйте следующую блок-схему:



**Рис.1.5**. Блок-схема линейного алгоритма нахождения суммы двух натуральных чисел.

**Выполнение задания №3.**

**Постановка задачи:**

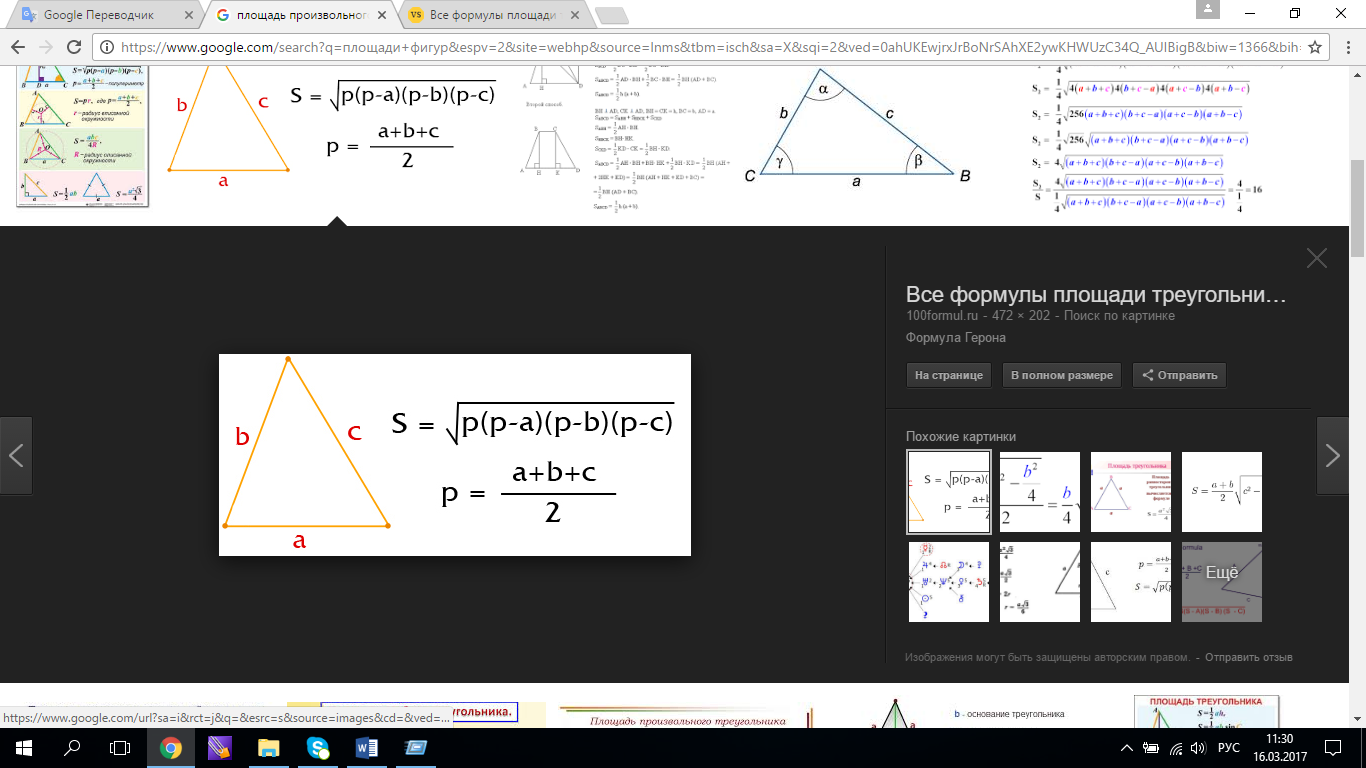
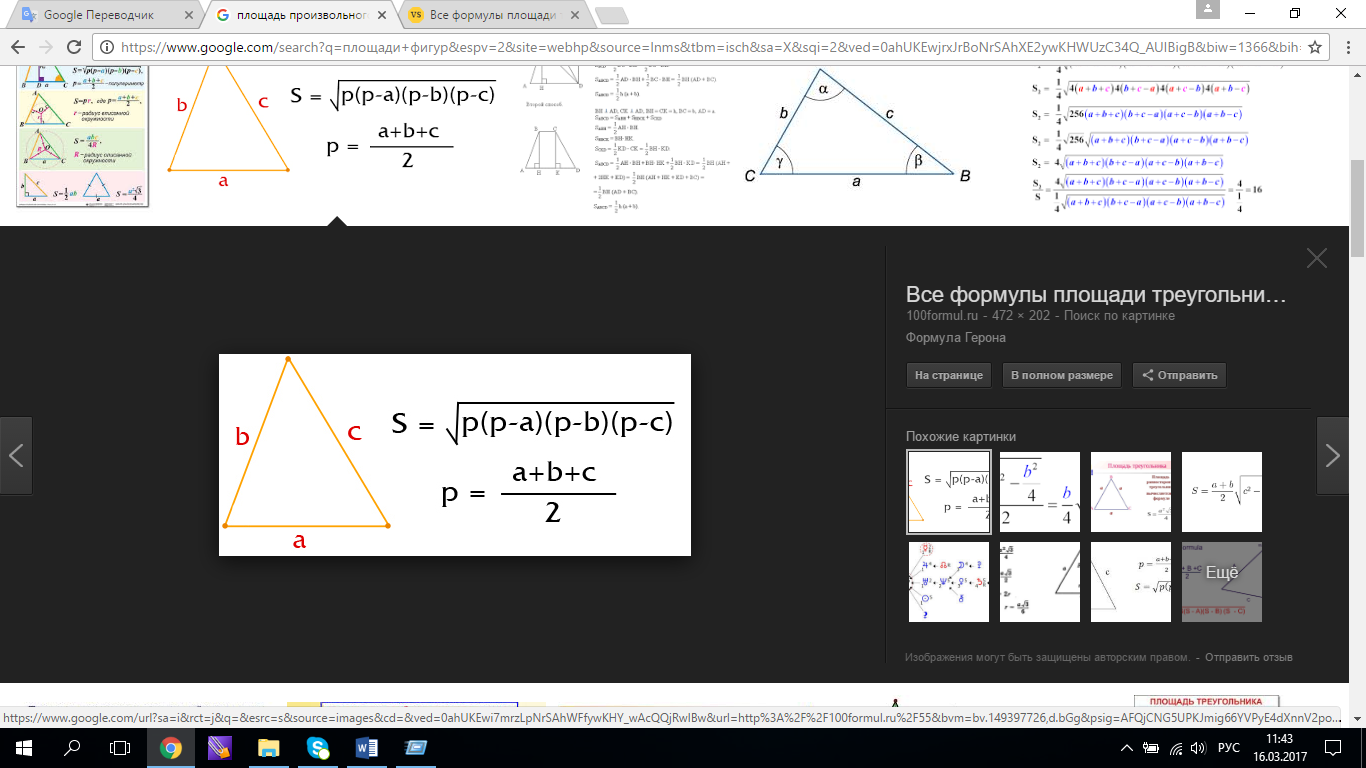
*Входные переменные:*

Веществ. A, B, C – длины сторон треугольника.

*Выходные переменные:*

Веществ. S – площадь треугольника*.*

***Примечание:***Если известны три стороны произвольного треугольника, то площадь вычисляется по следующей формуле:

, где - это полупериметр треугольника.

Так как по условию задачи нас интересует именно значение площади треугольника, то в качестве выходных данных указывается значение вычисленной площади *S*. Но для ее вычисления используется полупериметр *р.* И если мы оставим постановку задачи в исходном виде, то потеряем описание этой переменной, так как значение *р* нигде не указанно. Чтобы этого избежать, мы либо можем все-таки указать значение *р* в выходных данных, если это напрямую не противоречит условию, либо добавить в постановку задачи пункт *Промежуточные данные,* для того, чтобы мы могли указать все переменные, участвующие в промежуточных вычислениях. Тогда постановка задачи будет выглядеть следующим образом:

*Входные переменные:*

Веществ. a, b, c – длины сторон треугольника.

*Промежуточные переменные:*

Веществ. p – полупериметр треугольника.

*Выходные переменные:*

Веществ. S – площадь треугольника.

**Выполнение задания №4.**

* Если во вкладке *Вставка* нет *Редактора уравнений*, то открыть *Файл -> Параметры -> Настроить ленту -> Все команды -> Редактор уравнений -> Добавить*.
* Начертить блок-схему:



**Рис.1.6.** Блок-схема линейного алгоритма нахождения площади треугольника.

1. **Задания для самостоятельной работы:**
2. Ввести значение угла в градусах. Вычислить. предварительно переведя угол в радианы. Составить блок-схему алгоритма решения и постановку задачи.
3. Вычислить. Составить блок-схему алгоритма решения и постановку задачи.
4. Найти площадь трапеции, если четыре заданных числа задают длины ее сторон. Составить блок-схему алгоритма решения и постановку задачи.
5. Вычислить тригонометрические функции для угла A. Составить блок-схему алгоритма решения и постановку задачи.
6. Вычислить возраст человека, если известен год его рождения. Составить блок-схему алгоритма решения и постановку задачи.
7. Дана величина А, выражающая объем информации в байтах. Перевести А в более крупные единицы измерения информации. Составить блок-схему алгоритма решения и постановку задачи.
8. **Контрольные вопросы:**
9. Что такое алгоритм?
10. Какие существуют разновидности алгоритмов?
11. Какие существуют формы представления алгоритмов?
12. Что такое псевдокод?
13. Что такое постановка задачи и как ее составить?

# Лабораторная работа №2

## **Построение разветвляющихся алгоритмов**

**Цель работы**

Ознакомление студентов с алгоритмами с ветвлением. Построение псевдокода и блок-схем для таких алгоритмов.

1. **Теоретические сведения:**

*Разветвляющийся алгоритм* – это алгоритм, в котором есть структура ветвления.

*Ветвление* - это выбор действия в зависимости от выполнения какого-нибудь условия. Ветвление может быть *полным* или *неполным.*

Пример: Даны два числа, выбрать большее из них.

Решение: Пусть исходными данными являются переменные А и В. Их значения будут задаваться вводом. Значение большего из них должно быть присвоено переменной С и выведено на экран компьютера.

Например, если А = 5, В = 8, то должно получиться: С = 8.Поэтому можно записать следующий алгоритм:

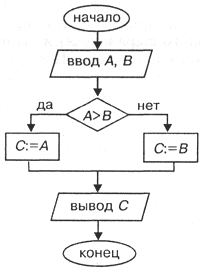
1. Задать численное значение А и В;

2. Проверить условие А > B;

3. Если условие выполняется, тогда присвоить переменной С значение переменной А;

4. Если условие не выполняется, тогда присвоить переменной С значение переменной В;

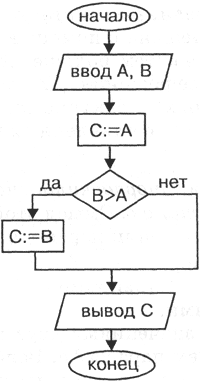
5. Написать ответ С (рис 2.1).



**Рис. 2.1.** Алгоритм выбора большего из двух чисел

(с полным ветвлением).

В алгоритме на (рис. 2.1) используется полное ветвление. Эту же самую задачу можно решить, применяя структурную команду неполного ветвления. Блок-схема такого алгоритма изображена на (рис. 2.2):



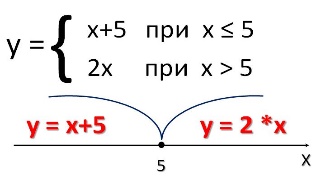
**Рис. 2.2.** Алгоритм выбора большего из двух чисел (с неполным ветвлением)

Алгоритм можно также описать псевдокодом.

*Псевдоко́д* — компактный (зачастую неформальный) язык описания алгоритмов, использующий ключевые слова языков программирования, но опускающий несущественные подробности и специфический синтаксис. Псевдокод обычно опускает детали, несущественные для понимания алгоритма человеком.

*Главная цель использования псевдокода* — обеспечить понимание алгоритма человеком, сделать описание более воспринимаемым, чем исходный код на языке программирования. Псевдокод широко используется в учебниках и научно-технических публикациях, а также на начальных стадиях разработки компьютерных программ.

1. **Задания к работе**
2. Построить блок-схему алгоритма решения следующей задачи:

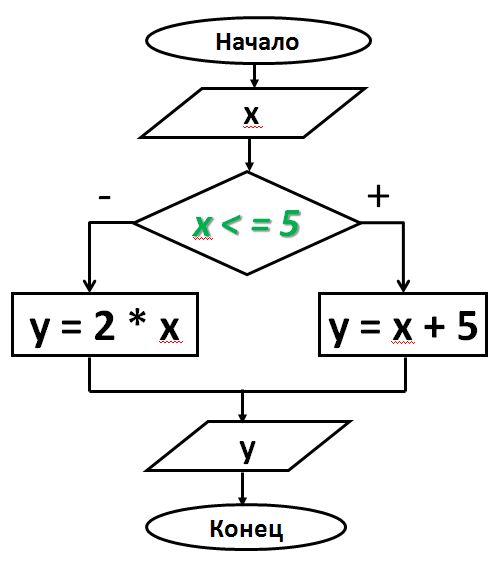


1. Составить псевдокод для решения задачи №1.
2. **Порядок выполнения лабораторной работы**

* Создайте новую рабочую книгу *MS Visio 2013* с названием «Условный алгоритм».

**Выполнение задания №1.**

* Путем перетаскивания необходимых блоков в рабочую область нарисуйте следующую блок-схему:



**Рис.2.3**. Блок-схема условного алгоритма решения

системы уравнений.

**Выполнение задания №2.**

* Во вкладке *Главная* выберите пункт *Текст,* разместите его в удобное для Вас место на рабочем листе и напечатайте псевдокод для указанной выше задачи:

алг Решение системы ур-й

арг цел x, y

нач

ввод x

если x <= 5 то

y = x + 5

иначе y = 2 \* x

все

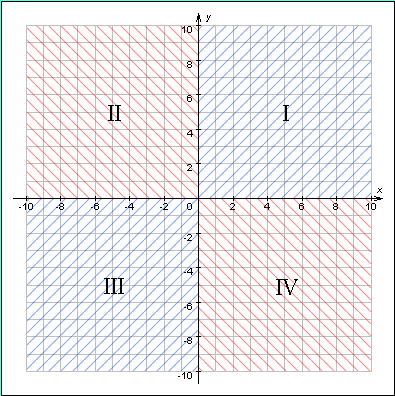
вывод y

кон

1. **Задания для самостоятельной работы:**

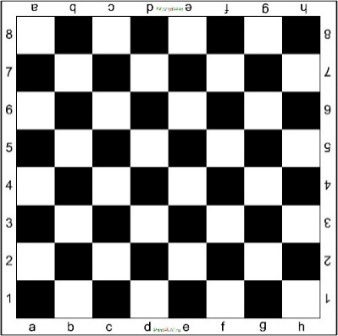
Для представленных задач составить псевдокод и построить блок-схемы.

1. Даны 3 вещественные числа *a, b, c*. Определить, имеется ли среди них хотя бы одна пара равных между собой чисел.
2. Даны 3 вещественных числа *a, b, c*. Определить, является ли треугольник со сторонами *а, b, c* равносторонним.
3. Определить, существует ли треугольник со сторонами *а, b, c*, если учесть, что треугольник существует только тогда, когда сумма двух любых его сторон больше третьей.
4. Определить, является ли число А делителем числа В или наоборот, является ли число В делителем числа А. Ответом должно служить сообщение "В является делителем А" или "А является делителем В".
5. Задан радиус круга и сторона квадрата. Определить, поместится ли круг в квадрат?
6. В магазине скидка в 3% предоставляется, если сумма покупки больше 200$. По заданной сумме покупки определить стоимость покупки.
7. Составить блок-схему и псевдокод на задачу, которая вычисляет оптимальный вес пользователя, сравнивает его с реальным весом и выдает рекомендации о необходимости поправиться или похудеть. Оптимальный вес вычисляется по формуле: рост (в сантиметрах) -100.
8. Определить, в какую из областей попадет точка с указанными координатами (x;y).



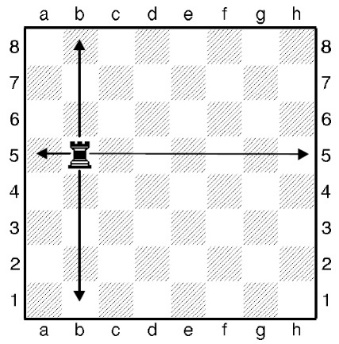
**Рис.2.4**. Система координат.

1. Поле шахматной доски определяется парой натуральных чисел, каждое из которых не превосходит 8. По введенным координатам двух полей выясните, имеют ли эти поля одинаковый или разный цвет.



**Рис.2.5**. Шахматная доска.

1. Дано целое четырехзначное число. Выяснить, является ли оно палиндромом (перевертышем, например, 2222,1221 и тд.).
2. Дано трехзначное число. Определить все трехзначные числа, которые можно получить из чисел данного числа.
3. Вводятся четыре координаты: координаты ладьи (два числа) и координаты другой фигуры (два числа). Координаты – целые числа от 1 до 8. Определить, сможет ли ладья за один ход побить другую фигуру. Вывести соответствующее сообщение.



**Рис.2.6**. Движение ладьи по шахматной доске.

1. Требуется определить принадлежность введенного числа данной арифметической прогрессии. Прогрессия задается двумя последовательными членами. В качестве руководства к составлению алгоритма используйте приведенные ниже словесный алгоритм к данной задачи:

***Начало алгоритма***

1. Введите два последовательных члена арифметической прогрессии.

2. Введите произвольное целое число.

3. Найдите разность (d) арифметической прогрессии.

4. Найдите разность между введенным числом и членом арифметической прогрессии.

5. Найдите остаток от деления нацело найденной разности на d.

6. Организуйте проверку остатка:

- если остаток от деления равен 0, выведите сообщение:

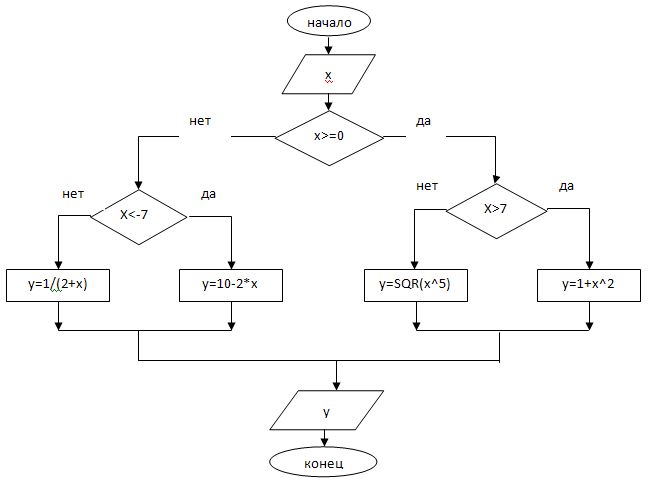
«Число принадлежит рассматриваемой арифметической прогрессии»;

- иначе выведите сообщение:

«Число не принадлежит рассматриваемой арифметической прогрессии».

***Конец алгоритма***

1. По изображенной блок-схеме составьте задачу и напишите псевдокод.



1. **Контрольные вопросы:**
2. Что такое разветвляющийся алгоритм?
3. Что такое полное ветвление?
4. Что такое неполное ветвление?
5. Что такое псевдокод?
6. Основная цель написания псевдокода?

# Лабораторная работа №3

## **Организация структуры ветвления посредством**

## **оператора *switch-case.***

**Цель работы**

Ознакомление студентов с оператором swich-case. Построение псевдокода и блок-схем для таких алгоритмов.

1. **Теоретические сведения:**

Очень часто в процессе написания программы требуется писать длинные if-else конструкции, например, когда мы получаем какой-либо ключ от пользователя. Конструкция switch-case — это удобная замена длинной if-else конструкции, которая сравнивает переменную с несколькими константными значениями.

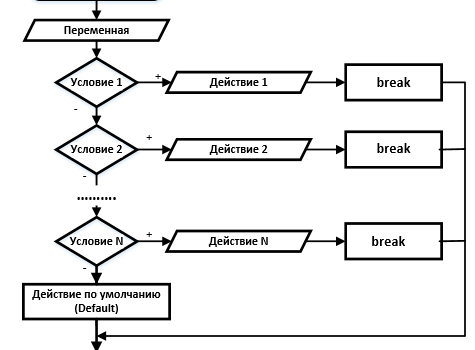
Оператор ветвления switch выполняется следующим образом:

* вводится (либо вычисляется) значение **Переменной** для оператора switch;
* полученное значение сравнивается с метками (**Константами**) в блоках case (Условие 1, Условие 2,...,Условие N), сравнение производится до тех пор, пока не будет найдена метка, соответствующая значению введенной переменной (либо вычисленному значению целочисленного выражения);

**Примечание:** *Константы в блоках case должны быть целого типа (могут быть символами).* *Все константные выражения должны иметь разные значения, но быть одного и того же целочисленного типа.*

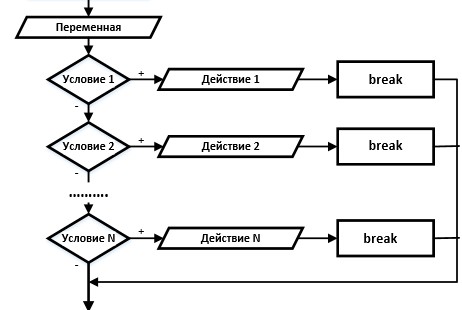
* выполняются те действия, которые соответствуют метке case;
* если соответствующая метка не найдена, то выполняются **Действия по умолчанию,** описанный в блоке default.

Действия по умолчанию – это альтернативные действия, которые будут выполняться в том случае, если ни одно из условий не будет давать значение «истинна».



**Рис.3.1**. Организация ветвления с действием по умолчанию.

Альтернатива default может отсутствовать, тогда не будет произведено никаких действий.



**Рис.3.2**. Организация ветвления без действия по умолчанию.

**Примечание**: Опция break осуществляет выход из оператора switch и переход к следующему за ним оператору. При отсутствии опции break будут выполняться все операторы, начиная с помеченного данной меткой и кончая оператором в опции default.

Ветвление, организованное с помощью оператора switch-case в псевдокоде будет выглядеть следующим образом:

Выбор

при условие 1: действие 1; выход;

при условие 2: действие 2; выход;

……………………………………..

при условие N: действие N; выход;

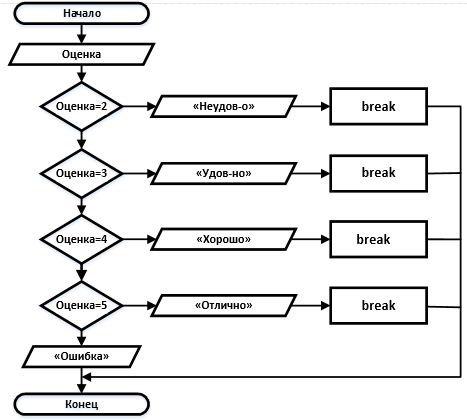
иначе действие

все

1. **Задания к работе**
2. Построить блок-схему для алгоритма решения следующей задачи: С клавиатуры вводится оценка студента в виде целого числа. В качестве результата работы программы выводится сообщение с символьным обозначением оценки. Если было введено некорректное значение, то выдается сообщение об ошибке.
3. Создать псевдокод для задачи №1.
4. **Порядок выполнения лабораторной работы**

**Выполнение задания №1.**

* Путем перетаскивания необходимых блоков в рабочую область нарисуйте следующую блок-схему:



**Рис.3.3**. Определение оценки.

**Выполнение задания №2.**

алг Определение оценки

арг цел Оценка

нач

ввод Оценка

Выбор

при Оценка = 2: вывод «Неудов-но»; выход;

при Оценка = 3: вывод «Удов-но»; выход;

при Оценка = 4: вывод «Хорошо»; выход;

при Оценка = 5: вывод «Отлично»; выход;

иначе вывод «Ошибка»

все

конец

1. **Задания для самостоятельной работы:**

Написать псевдокод и построить блок-схемы для следующих задач:

1. С клавиатуры вводятся 2 целых числа A, B и знак операции (+, -, \*, /). В зависимости от знака операции выполнить соответствующее действие.
2. Найти корни квадратного уравнения вида ax2 + bx + c = 0 (a ≠ 0). В случае, если корней нет, то выдать соответствующее сообщение.
3. С клавиатуры вводятся порядковый номер дня недели. В соответствии с веденым значением вывести символьное название дня недели.
4. Составить расписание на неделю. Пользователь вводит порядковый номер дня недели и у него на экране отображается, то, что запланировано на этот день.
5. Составить программу, в которой реализуются следующие условия: в соответствии с введенным числом вывести таблицу умножения с его участием.
6. **Контрольные вопросы:**
7. Когда применяется структураswitch-case?
8. Что такое значение по умолчанию?
9. Для чего используется оператор break?
10. Какие операторы будут выполняться, если не использовать break?
11. Могут ли константы блока case иметь символьные значения?

# Лабораторная работа №4

## **Построение циклических алгоритмов**

**Цель работы**

Ознакомление студентов с циклическими алгоритмами. Построение псевдокода и блок-схем для таких алгоритмов.

1. **Теоретические сведения:**

*Циклическим* алгоритмом называют алгоритм, в котором набор некоторых действий должен неоднократно выполняться.

Циклические алгоритмы по способу организации выхода из цикла можно разделить на два типа:

* арифметические (циклы со счетчиком);
* итерационные (циклы с условием).

Количество повторений в первых (циклы со счетчиком) заранее известно или может быть легко вычислено. Количество повторений во-вторых (циклы с условием) - заранее неизвестно. Выход из таких циклов осуществляется в зависимости от истинности условия, которое определяет дальнейшую работу цикла: будет ли он выполняться еще раз или будет завершен. Для успешного завершения цикла среди действий внутри цикла должны производиться операции, влияющие на условие. Кроме этого, условие должно состоять из корректных выражений и значений. Иначе цикл может стать бесконечным, что приведет к ошибке.

***Алгоритмы цикла типа «FOR» / «ДЛЯ» (цикл со счетчиком).***

Цикл, в котором количество повторения действий известно заранее, называется **циклом со счетчиком** (или **параметром).**

**Как работает цикл?**

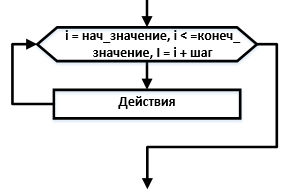
Сначала счетчику (параметру цикла) присваивается начальное значение, затем проверяется выполнение условия: значение параметра цикла меньше или равно конечному значению. Если условие истинно, то выполняется тело цикла (команды, идущие после начала цикла) и значение параметра цикла автоматически увеличивается на единицу (или на любой другой шаг, который вы указали). Далее опять проверяется тоже самое условие. Проверка продолжается до тех пор, пока условие не станет ложным.

Если условие будет ложно на самом первом шаге, то операторы тела цикла не выполняется ни разу.

Правила записи оператора цикла со счетчиком:

1. Определить параметр цикла (изменяющуюся величину, связанную с повторяющимися действиями) и дать ей имя.
2. Определить начальное и конечное значения параметра цикла.
3. Определить повторяющиеся действия и записать их.

На блок-схеме это выглядит как на рис. 4.1.



**Рис 4.1.** Блок-схема циклического алгоритма For

***Алгоритм цикла типа «WHILE» / «ПОКА».***

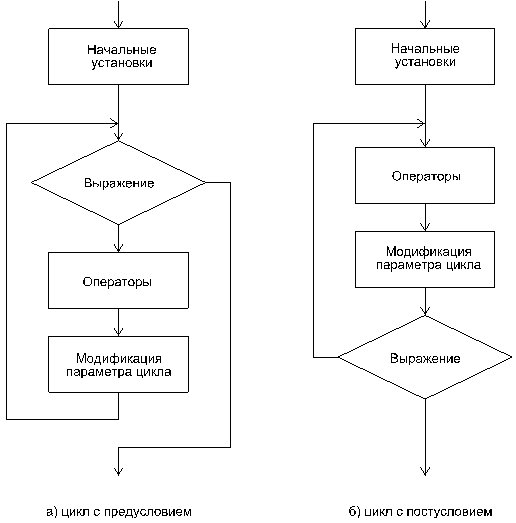
Часто приходиться решать задачи, число повторений в которых заранее не известно и выявляется в процессе выполнения некоторого количества повторений тела цикла. В этом случае применяется цикл с условием.

В одной из разновидностей цикла условие проверяется перед выполнением цикла. Поэтому данное условие правильно будет назвать *условием продолжения цикла.* Цикл такого вида называется ***циклом с предусловием***.

Проверка условия выполнения цикла производится в самом начале – если значение условия истинно, то выполняется тело цикла и снова вычисляется выражение условия выполнения цикла. Тело цикла будет выполнятся до тех пор, пока проверка этого условия будет давать результат «истина», то есть пока условие выполняется.

Если же условие проверяется после выполнения тела цикла, то данное условие правильно будет назвать *условием выхода из цикла*. Цикл такого вида называется ***циклом с постусловием***.

Особенностью этого вида цикла является то, что тело цикла выполняется хотя бы один раз.



**Рис 4.2.** Разновидности циклического алгоритма While.

***Цикл с предусловием (WHILE).***

 Порядок выполнения оператора цикла с предусловием:

1. Сначала проверяется условие, если оно верно, то выполняется оператор;

2. Затем опять проверяется условие и снова выполняется оператор, пока условие не перестанет выполняться;

3. Если условие не верно, то оператор игнорируется и управление передается следующему за циклом оператору.

***Цикл с постусловием (DO WHILE).***

 Порядок выполнения оператора цикла с постусловием:

1. Сначала выполняется оператор;

2. Затем проверяется условие. Если оно истинно, то снова выполняется оператор, пока условие не перестанет выполняться;

3. Если условие не верно, то оператор игнорируется и управление передается следующему за циклом оператору.

1. **Задания к работе**
2. Построить блок-схему для алгоритма решения задачи нахождения суммы 5 произвольных чисел c помощью цикла For.
3. Составить псевдокод для решения задачи №1.
4. Построить блок-схему для алгоритма решения задачи нахождения суммы 5 произвольных чисел c помощью цикла c предусловием.
5. Составить псевдокод для решения задачи №3.
6. Построить блок-схему для алгоритма решения задачи нахождения суммы 5 произвольных чисел c помощью цикла c постусловием.
7. **Порядок выполнения лабораторной работы**

* Создайте новую рабочую книгу *MS Visio 2013* с названием «Циклический алгоритм».

**Выполнение задания №1.**

Давайте определим, как найти сумму чисел? Надо взять первые два числа и сложить их, потом к полученной сумме прибавить следующее число и так далее пока не сложим все числа.

Заметим, что всегда повторяется одно и то же действие: к старой сумме прибавляется следующее число.

Только самое первое действие немного отличается. Мы складываем два числа. Было бы хорошо, если бы все действия были бы одинаковыми, тогда задачу можно было бы решить с помощью цикла. Этого можно добиться, если задать начальную сумму, равной нулю. (Символично это будет первое число, и оно не изменит значение суммы). А затем к ней прибавим первое число, к полученной сумме второе и так далее.

Мы получили алгоритм нахождения суммы чисел. Этот алгоритм можно использовать во многих задачах. Его необходимо понять и запомнить.

Обозначим сумму S, а число переменной X. Тогда алгоритм нахождения суммы 5 чисел будет иметь следующий вид:

* 1. Параметр цикла: счетчик, который считает, сколько чисел уже введено и прибавлено к сумме. Пусть его имя будет i.
  2. Начальное значение параметра цикла равно 0, конечное значение

параметра цикла равно 5.

* 1. Повторяющиеся действия:

|  |  |
| --- | --- |
| Ввод X | на каждом новом шаге мы вводим значение следующего числа для прибавления. |
| S = S + X | новое значение суммы S (стоящее слева) получается путем прибавления к старому значению суммы S (стоящее справа) только что введенного числа X. |

Имейте ввиду, что начальное значение суммы S=0.

Математическая модель, описывающая формальную схему работы цикла со счетчиком:

* + 1. S = 0.
    2. Действия цикла:
       - 1. Параметр цикла – переменная i.
         2. Начальное значение i равно 1, конечное значение i равно 5.
         3. Повторяющиеся действия:

*Ввод Х*

*S=S+X*

* + 1. После завершения работы цикла:

*Вывод S*

* Путем перетаскивания необходимых блоков в рабочую область создайте следующую блок-схему:

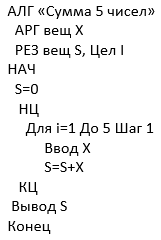


**Рис.4.3**. Нахождение суммы 5 произвольных чисел. Цикл For.

**Выполнение задания №2.**

Решение этой задачи схоже с решением задачи №2 в Лабораторной работе №4, только вместо сложение обычных чисел, здесь необходимо сложить именно элементы массива. Общий вид блок схемы и псевдокода схожи.

• Во вкладке *Главная* выберите пункт *Текст* и разместите его в удобное для Вас место на рабочем листе и напечатайте псевдокод для описанной выше задачи:



**Выполнение задания №3.**

Так как мы будем использовать цикл с предусловием, то инициализируем наш счетчик i = 1 до начала проверки условий и тела цикла.

**Примечание**: Здесь инициализация начального значения счетчика обязательна, если мы не инициализируем счетчик цикла, то в него попадет «мусор» и компилятор в лучшем случае выдаст нам ошибку.

Затем мы описываем условие цикла — «пока переменная i меньше или равна 5 — выполняй цикл». При каждой итерации цикла значение переменной-счетчика i увеличивается на единицу внутри цикла.

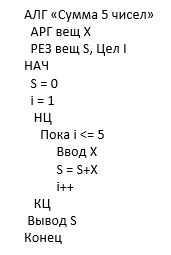
* Путем перетаскивания необходимых блоков в рабочую область создайте следующую блок-схему:



**Рис.4.4**. Нахождение суммы 5 произвольных чисел. Цикл While.

**Выполнение задания №4.**

**Примечание:** В циклических алгоритмах с условием в псевдокоде блок условия не обозначается ключевым словом *Если.* Так как тело цикла выполняется только пока условие истина, то для обозначения условия используется ключевое слово *Пока.*

****

**Выполнение задания №5.**



**Рис.4.5**. Нахождение суммы 5 произвольных чисел. Цикл Do While.

1. **Задания для самостоятельной работы:**

Написать псевдокод и построить блок-схемы для следующих задач:

1. Найти сумму чисел, кратных трем, в диапазоне от 0 до 50.
2. Найти сумму первых десяти чисел, кратных пяти.
3. Найти произведение четных чисел в диапазоне от 2 до 30.
4. Вводятся положительные числа. Прекратить ввод, когда сумма введенных чисел превысит 100.
5. Требуется найти сумму чисел, кратных 7, в диапазоне от 0 до 100. Вывести сумму и их количество.
6. Вводятся n чисел. Определите количество отрицательных, положительных чисел и нулей.
7. Вывести такие двузначные числа, которые делятся на 4, но не делятся

на 6.

1. Определить количество целых чисел, кратных 3 (от 3 и далее),

дающих в сумме число, не превышающее 200.

1. Вводятся 10 чисел. Вывести на экран сумму положительных и отрицательных чисел и их количество.
2. Определить значения произведений чисел а и b. Числа a изменяются

от 1 до 11 с шагом 1, b – от 1 до 3 с шагом 0,2.

1. Решив заняться легкой атлетикой, вы пробежали в первый день 2 км. Сколько км Вы пробежите за 2 недели, если каждый день вы увеличиваете дистанцию на 10% от предыдущего дня.
2. **Контрольные вопросы:**
3. Что такое циклический алгоритм и когда он применяется?
4. Что представляет из себя счетчик с параметром?
5. Что такое параметр цикла?
6. Что такое итерационные алгоритмы?
7. Циклы с предусловием/постусловием. Отличия?

# Лабораторная работа №5

## **Одномерные массивы**

**Цель работы**

Ознакомление студентов с одномерными массивами. Построение

псевдокода и блок-схем для таких алгоритмов.

1. **Теоретические сведения:**

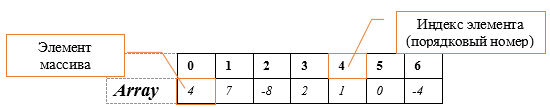
**Массив** – это некоторая совокупность ячеек, объединенная одним обозначением (массивом может быть одна ячейка). Иначе говоря, это последовательность однотипных элементов, имеющих общее имя, причем каждый элемент этой последовательности определяется порядковым номером (индексом) элемента. Индекс определяет позицию элемента в массиве. С помощью индекса осуществляется доступ к отдельному элементу массива. Первый элемент массива имеет нулевой индекс.

Всякий массив обязательно имеет размерность. Массивы

бывают одномерными, двумерными и т. д.

**Одномерный массив** представляет собой линейку или столбец символов.

Его также называют *вектором.*



**Рис.5.1.** Одномерный массив.

Перед использованием массив обязательно должен быть описан: заданы тип данных массива, имя массива и его размерность.

На**рис.5.1***.* изображен массив Array, который состоит из 7 целочисленных значений. Его описание будет выглядеть следующим образом: **Int Array [7].**

В результате, в памяти компьютера выделяется 7 ячеек типа int. Если

одна ячейка занимает 2 байта, то всего будет выделено 14 байт памяти.

Чтобы получить доступ к отдельному элементу массива по его индексу,

нужно после имени массива в квадратных скобках указать номер этого элемента.

***Инициализация массива.***

Инициализация массива (или переменной) означает присвоение значения в строке объявления. Список элементов массива должен быть заключён в фигурные скобки {}.  
 Количество элементов массива должно равняться размеру массива. Если Вы предоставите меньше элементов, то последние элементы массива будут проинициализированы нулями. Если больше - то лишние элементы будут проигнорированы

Общий вид инициализации с заданием размера массива:

тип имя\_массива[размер] = {список\_значений},

тогда для нашего массива инициализация будет выглядеть следующим образом:

Int Array[7] = {4, 7, -8, 2, 1, 0, -4}.

Можно также присвоить значения элементам массива по индексам:

Int Array[7];

Array[0] = 4;

Array[1] = 7;

Array[2] = -8;

Array[3] = 2;

Array[4] = 1;

Array[5] = 0;

Array[6] = -4;

***Заполнение массива.***

В том случае, если при объявлении массива была указана только

его размерность без инициализации элементов массива, то заполнить

массив можно тремя основными способами:

1. Ввести элементы массива с клавиатуры;
2. С помощью оператора присваивания (по формуле или с

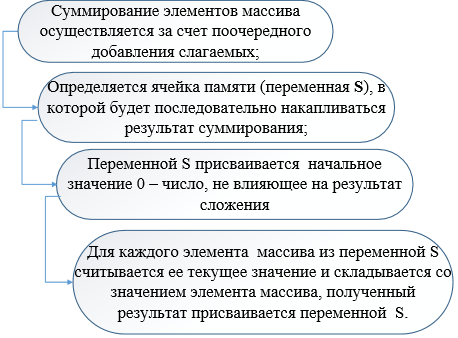
помощью генератора случайных чисел)

1. Считать элементы массива из отдельного файла.
2. **Задания к работе**
3. Найти сумму элементов массива а[10]. Построить блок- для алгоритма решения задачи.
4. Написать псевдокод для задачи №1.
5. **Порядок выполнения лабораторной работы**

* Создайте новую рабочую книгу *MS Visio 2013* с названием «Одномерные массивы».

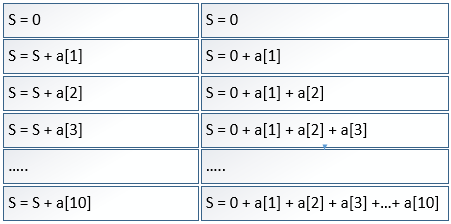
**Выполнение задания №1.**

Давайте разберемся, как суммируются элементы массива:



**Рис.5.2.** Алгоритм суммирования элементов массива.

Тогда на каждой итерации (шаге) выполнения программы происходят следующие вычисления:

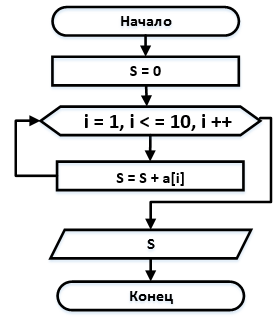


**Рис.5.3.** Математическая модель алгоритма суммирования элементов массива.

Так как размерность массива нам заранее известна из условий задачи, то

для работы с элементами массива будем использовать цикл For.

* Откройте Ф*игуры простой блок-схемы* в наборе элементов и изобразите следующую блок-схему:



**Рис.5.4.** Сумма элементов массива.

***Примечание:*** Обратите внимание на то, что здесь массив начинается

с единицы, а не с 0. В этом случае следует помнить, что тогда конечной

точкой будет не значение 9, а 10 включительно.

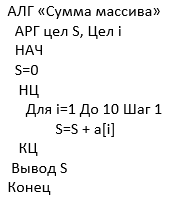
**Выполнение задания №2.**

Решение этой задачи схоже с решением задачи №2 в Лабораторной работе

№4, только вместо сложение обычных чисел, здесь необходимо сложить именно элементы массива. Общий вид блок схемы и псевдокода схожи.

* Во вкладке *Главная* выберите пункт *Текст* и разместите его в удобное

для Вас место на рабочем листе и напечатайте псевдокод для описанной выше задачи:

****

1. **Задания для самостоятельной работы:**

Написать псевдокод и построить блок-схемы для следующих задач:

1. В массиве а(10) определить кол-во нулей.
2. В массиве d(15) найти сумму чисел, расположенных на нечетных

местах.

1. В массиве чисел с(10) найти кол-во чисел, попавших в интервал [a , b].
2. Найти номер первого нулевого элемента массива х(20) и сумму

элементов, предшествующих ему.

1. Известно, что в массиве а(16) кол-во отрицательных элементов равно

кол-ву положительных. Составить новый массив так, чтобы

чередовались положительные и отрицательные числа.

1. Дана последовательность чисел с(16). Найти произведение элементов

этой последовательности до первого нулевого и сумму элементов, расположенных после него.

1. Из данного массива чисел х(15) исключить последнее положительное число. Оставшиеся положительные числа переписать в новый массив.
2. Все положительные элементы массива а(10), расположенные правее первого нулевого элемента, увеличить в два раза.
3. Из заданного массива а(12), не содержащего нулей, получить массив b(12), приняв в качестве первых его элементов все положительные элементы массива а с сохранением порядка их следования, а в качества остальных элементов все отрицательные элементы так же с сохранением порядка их следования.
4. Дан массив с(15) состоящий из нулей и единиц. Подсчитать кол-во 0 и 1,

и кол-во нулей до первой единицы.

1. По данным числам x, y, z получить массив а(17), где а(1)=х, а(2)=у,

а(3)=x, а каждый следующий элемент массива определяется как среднее арифметическое трех предшествующих.

1. Все элементы массива а(9), начиная с первого по порядку положительного элемента, уменьшить на 0.5, если значение превышает 1, и увеличить на 0.5 в противном случае.
2. Если наименьший элемент массива х(27) больше 0.1, то все положительные элементы массива заменить единицей.
3. Среди отрицательных элементов массива х(50) найти минимальный и помножить на него все отрицательные элементы, стоящие левее этого минимального.
4. Из отрицательных элементов массива х(30) расположенных левее минимального элемента, сформировать новый массив.
5. В массиве у(25) поменять местами минимальный элемент с первым положительным элементом.
6. **Контрольные вопросы:**
7. Что такое массив?
8. Что такое одномерный массив?
9. Что такое индекс массива?
10. Как можно получить доступ к элементу массива?
11. Что такое инициализация массива?
12. Каким образом можно заполнить массив?
13. Может ли массив одновременно содержать и целые и вещественные элементы?

# Лабораторная работа №6

## **Двумерные массивы**

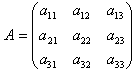
**Цель работы**

Ознакомление студентов с двумерными массивами. Построение псевдокода и блок-схем для таких алгоритмов.

1. **Теоретические сведения:**

При решении практических задач часто приходится иметь дело с различными таблицами данных, математическим эквивалентом которых служат матрицы. Такой способ организации данных, при котором каждый элемент определяется номером строки и номером столбца, на пересечении которых он расположен, называется *двумерным массивом*.

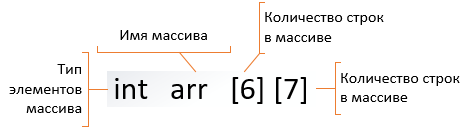
Например, дана матрица размерностью 3\*3, то есть в ней будет три строки, а в каждой строке по три элемента.



**Рис.6.1**. Квадратная матрица.

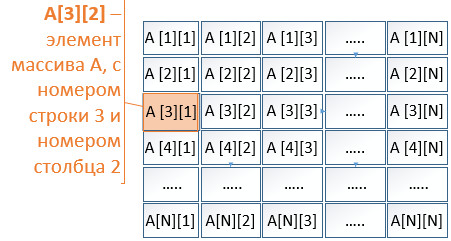
Их можно занести в память компьютера, используя понятие двумерного массива. Положение элемента в массиве определяется двумя индексами. Они показывают номер строки и номер столбца.

Индексы записываются в квадратных скобках каждый (или же могут разделяться запятой):



**Рис.6.2**. Объявление двумерного массива.

Доступ к элементу массива осуществляется по номеру строки и столбца соответственно:



**Рис.6.3**. Двумерный массив.

В блок-схеме работа с двумерными массивами осуществляется с помощью вложенных циклов.

***Вложенные циклы*** – это, когда определённый цикл повторяется многократно в другом цикле, охватывающем данный. Вложенный цикл должен полностью находиться в области внешнего цикла. Вложенный цикл может быть один, но может быть и несколько вложенных циклов. Второй вложен в первый, третий – во второй и т.д. Ниже с помощью псевдокода представлена структура циклического алгоритма, содержащего несколько вложенных циклов:

Начало цикла 1

Начало цикла 2

Начало цикла 3

Тело цикла 3

Конец цикла 3

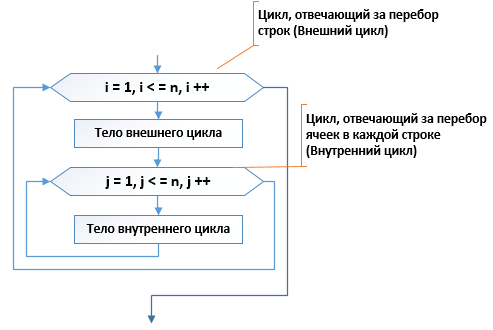
Конец цикла 2

Конец цикла 1

Для организации и внутреннего, и внешнего циклов могут использоваться разные типы алгоритмических структур (цикл с параметром, цикл с предусловием, цикл с постусловием).

На **рис.6.4.** представлена блок-схема алгоритма с внутренним циклом.

В данном случае и внешний и внутренний циклы организованы на базе алгоритмической структуры «цикл с параметром».



**Рис.6.4**. Вложенный цикл.

На каждом шаге внешнего цикла внутренний цикл выполняется несколько раз. Количество внутренних циклов на каждом внешнем цикле зависит от параметра внутреннего цикла.

Пусть, например, задано, что параметр внешнего цикла меняется от 1 до 5 с шагом 1, а параметр внутреннего цикла – от 1 до 10 с шагом 1.

Это означает, что на каждом шаге по внешнему циклу внутренний цикл будет выполняться 10 раз. Так как внешний цикл должен выполниться 5 раз, то внутренний цикл выполнится при этом 50 раз.

Типичная ошибка, когда в качестве счетчиков вложенных циклов (i и j в приведенном примере) используется одна и та же переменная. То есть нельзя в каждом из циклов использовать одну переменную i. Помнить об этом особенно важно, поскольку данная ошибка не обнаруживается на этапе компиляции. Ваша программа запустится, но делать будет вовсе не то, что вы от нее ждете. В приведенном примере (если допустить ошибку, заменив переменную j на i) внешний цикл выполнится всего 1 раз вместо 5-и. Возможна также ситуация, когда такая ошибка приведет к зацикливанию: внешний цикл будет выполняться бесконечно долго – программа зависнет.

Инициализация двумерного массива подобна инициализации одномерного массива.

1. **Задания к работе**
2. Дан целочисленный двумерный массив, размерности n х m, найти сумму всех элементов массива. Построить блок-схему алгоритма решения данной задачи.
3. Составить псевдокод для решения задачи №1.
4. **Порядок выполнения лабораторной работы**

* Создайте новую рабочую книгу *MS Visio 2013* с названием «Двумерный массив».

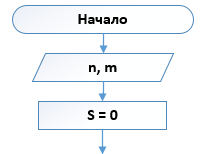
**Выполнение задания №1.**

Давайте определим, как найти сумму всех элементов двумерного массива? Алгоритм нахождения суммы элементов двумерного массива аналогичен нахождению суммы элементов одномерного массива, рассмотренного в предыдущей лабораторной работе, но используем не один индекс для указания адреса элемента, а два, для строк и столбцов.

Так как по условию задачи дан двумерный массив произвольной

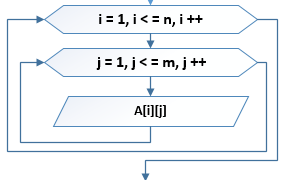
размерности n х m, то значение n – количество строк и m – количество столбцов, мы будем вводить с клавиатуры. А начальное значение суммы S нужно сделать равным 0, чтобы не повлиять на

решение (рис.6.5):



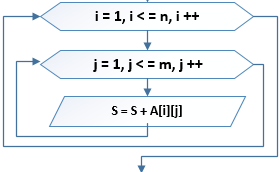
**Рис.6.5**. Ввод размерности массива.

Элементы массива будем вводить с клавиатуры. Чтобы ввести элементы, организуем вложенный цикл. Внешний цикл от 1 до n (кол-ва строк), внутренний цикл от 1 до m (кол-ва столбцов). Заполнение массива будет идти построчно: т.е. сначала берется первая строка и полностью заполняется элементами (количество элементов зависит от указанного кол-ва столбцов); затем вторая строка заполняется элементами и тд.



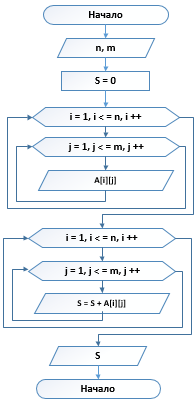
**Рис.6.6**. Ввод элементов массива.

Для нахождения суммы элементов будем использовать такой же цикл:



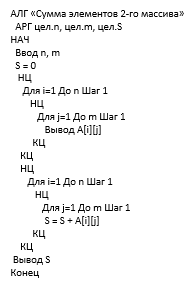
**Рис.6.7**. Определение суммы элементов массива.

В итоге должна получиться следующая блок-схема:



**Рис.6.8**. Суммы элементов массива.

**Выполнение задания №2.**



1. **Задания для самостоятельной работы:**

Написать псевдокод и построить блок-схемы для следующих задач:

1. Дан целочисленный двумерный массив, размерности n х m. Найти сумму и произведение всех элементов массива.
2. Дан целочисленный двумерный массив, размерности n х m. Найти сумму и произведение четных элементов.
3. Дан целочисленный двумерный массив, размерности n х m. Найти сумму и произведение элементов, кратных 3 и 5.
4. Дан целочисленный двумерный массив, размерности n х m. Найти количество отрицательных элементов, больше -9.
5. Дан целочисленный двумерный массив, размерности n х m. Найти номера нечетных элементов, стоящих на четных местах.
6. Дан целочисленный двумерный массив, размерности n х m. Найти максимум и минимум. Поменять их местами.
7. Дан целочисленный двумерный массив, размерности n х m. Заменить все элементы на их квадраты.
8. Дан целочисленный двумерный массив, размерности n х m. Найти среднее арифметическое всех элементов массива.
9. Дан целочисленный двумерный массив, размерности n х m. Выяснить, какое число встречается в какой строке раньше — положительное или отрицательное.
10. Дан целочисленный двумерный массив, размерности n х m. Выяснить, в какой строке последовательность является возрастающей или убывающей.
11. Дан целочисленный двумерный массив, размерности n х m. Вывести его элементы, индексы которых являются степенями двойки (1, 2, 4, 8, 16, ...).
12. Дан целочисленный двумерный массив, размерности n х m. Вывести на экран элементы, которые являются квадратами какого-либо числа.
13. Дан целочисленный двумерный массив, размерности n х m. Поменять местами первый и последний элементы.
14. Дан целочисленный двумерный массив, размерности n х m. Сформировать новый массив, состоящий из противоположных соответствующих элементов.
15. Дан целочисленный двумерный массив, размерности n х m. Вывести на экран те элементы, у которых остаток от деления на m равен k.
16. Вводятся результаты контрольной работы 10 учащихся. Определите число не удовлетворительных, удовлетворительных, хороших, отличных оценок. Вывести среднюю оценку, полученную учащимися за контрольную работу.
17. Ввести оценки N учеников по K предметам. Определить и вывести на экран количество учеников, не получивших ни одной "5".
18. В группе учится N студентов, студенты получили по четыре отметки за экзамен. Определить количество неуспевающих студентов и средний балл группы.
19. Дан целочисленный двумерный массив, размерности n х m. Сложить соответствующие элементы.
20. **Контрольные вопросы:**
21. Что такое двумерный массив?
22. Как получить доступ к элементам двумерного массива?
23. Как инициализируются элементы двумерного массива?

# Лабораторная работа №7

## **Создание тестовых вариантов**

**Цель работы**

Ознакомление студентов с основными понятиями, целями и задачами тестирования программного обеспечения и с созданием тестовых вариантов.

1. **Теоретические сведения:**

**Тестирование** — процесс выполнения программы с целью обнаружения ошибок.

Каждый тест определяет:

* свой набор исходных данных и условий для запуска программы;
* набор ожидаемых результатов работы программы.

Другое название теста — **тестовый вариант.** Полную проверку программы гарантирует *исчерпывающее тестирование.* Оно требует проверить все наборы исходных данных, все варианты их обработки и включает большое количество тестовых вариантов. Увы, но исчерпывающее тестирование во многих случаях остается только мечтой — срабатывают ресурсные ограничения (прежде всего, ограничения по времени).

Хорошим считают тестовый вариант с высокой вероятностью обнаружения еще не раскрытой ошибки. Успешным называют тест, который обнаруживает до сих пор не раскрытую ошибку.

*Пример из жизни. Проведем аналогию посещения врача больным.*

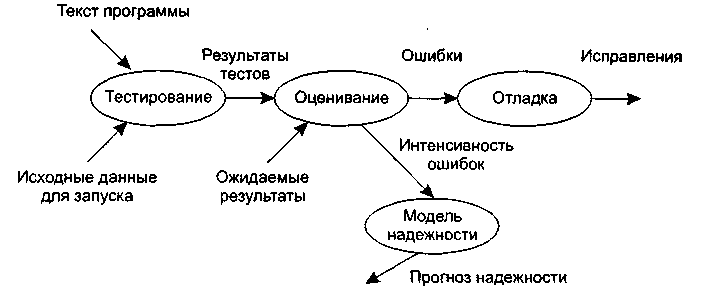
Если рекомендованное врачом лабораторное исследование не обнаружило причины болезни, не назовем же мы такое исследование удачным – оно неудачно: ведь счет пациента сократился на некоторую сумму, а он все так же болен. Если же исследование показало, что у больного, например, язва желудка, то оно является удачным, поскольку врач может прописать необходимый курс лечения. Аналогия здесь заключается в том, что программа, которую предстоит тестировать, подобна больному пациенту.

Что может тестирование?

Тестирование обеспечивает:

* обнаружение ошибок;
* демонстрацию соответствия функций программы ее назначению;
* демонстрацию реализации требований к характеристикам программы;
* отображение надежности как индикатора качества программы.

А чего не может тестирование? Тестирование не может показать отсутствия дефектов (оно может показывать только присутствие дефектов). Важно помнить это утверждение при проведении тестирования.



**Рис. 7.1.** Информационные потоки процесса тестирования

На входе процесса тестирования три потока:

* текст программы;
* исходные данные для запуска программы;
* ожидаемые результаты.

Выполняются тесты, все полученные результаты оцениваются. Это значит, что реальные результаты тестов сравниваются с ожидаемыми результатами. Когда обнаруживается несовпадение, фиксируется ошибка — начинается отладка (исправление).

После сбора и оценивания результатов тестирования начинается отображение качества и надежности ПО.

В конечном счете, если тесты не обнаруживают ошибок, появляется сомнение в том, что тестовые варианты достаточно продуманы и что в ПО нет скрытых ошибок. Такие ошибки будут, в конечном итоге, обнаруживаться пользователями и корректироваться разработчиком на этапе сопровождения (когда стоимость исправления возрастает в 60-100 раз по сравнению с этапом разработки).

Существуют 2 принципа тестирования программы:

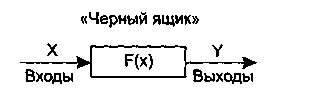
* функциональное тестирование (тестирование «черного ящика»);
* структурное тестирование (тестирование «белого ящика»).

### Тестирование «черного ящика».

**Известны:** функции программы.

**Исследуется:** работа каждой функции на всей области определения.

Как показано на рис. 7.2, основное место приложения тестов «черного ящика» — интерфейс ПО.



**Рис. 7.2.** Тестирование «черного ящика»

Эти тесты демонстрируют:

* как выполняются функции программ;
* как принимаются исходные данные;
* как вырабатываются результаты;
* как сохраняется целостность внешней информации.

При тестировании «черного ящика» рассматриваются системные характеристики программ, игнорируется их внутренняя логическая структура. Исчерпывающее тестирование, как правило, невозможно. Например, если в программе 10 входных величин и каждая принимает по 10 значений, то потребуется 1010 тестовых вариантов. Отметим также, что тестирование «черного ящика» не реагирует на многие особенности программных ошибок.

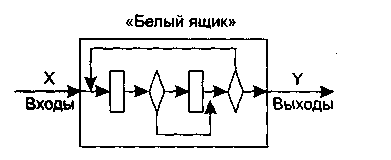
Стратегии «черного ящика»:

1. Тестирование функций.
2. Тестирование классов входных данных.
3. Тестирование классов выходных данных.
4. Тестирование области допустимых значений (тестирование границ класса).
5. Тестирование длины набора данных.
6. Тестирование упорядоченности набора данных.

### Тестирование «белого ящика».

**Известна:** внутренняя структура программы.

**Исследуются:** внутренние элементы программы и связи между ними



**Рис. 7.3.** Тестирование «белого ящика»

Объектом тестирования здесь является не внешнее, а внутреннее поведение программы. Проверяется корректность построения всех элементов программы и правильность их взаимодействия друг с другом. Тестирование по принципу «белого ящика» характеризуется степенью, в какой тесты выполняют или покрывают логику (исходный текст) программы. Исчерпывающее тестирование также затруднительно.

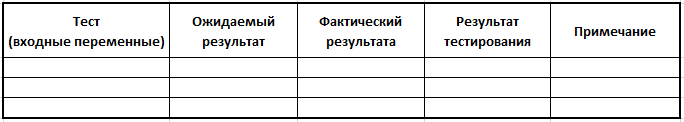
Стратегии «белого ящика»:

1. покрытие операторов;
2. покрытие решений;
3. покрытие условий;
4. покрытие решений/условий;
5. комбинаторное покрытие условий.

*На практике лучше использовать обе стратегии, так как это увеличит вероятность обнаружения ошибки.*

1. **Задания к работе**
2. Даны длины треугольника, определить вид треугольника и его площадь. Построить блок-схему и составить тестовые варианты для данной задачи.
3. **Порядок выполнения лабораторной работы**

* Открыть *Пуск -> Microsoft Office -> Excel -> Пустая книга.*
* Создать таблицу для тестового варианта со следующими полями:

****

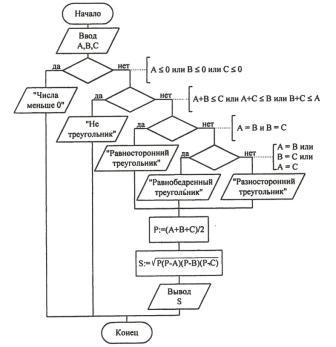
**Рис.7.4.** Общий вид таблицы для тестового варианта.

**Выполнение задания №1.**

Задача решается просто: вид треугольника определим, сравнивая стороны, а площадь вычислим по формуле Герона. Задание "выполнить контроль вводимых чисел" означает, что программа в случае ввода чисел, которые не могут интерпретироваться как стороны треугольника, должно выдаваться сообщение

об ошибках данных. Три числа нельзя интерпретировать как стороны треугольника, если хотя бы одно из них меньше или равно 0 или сумма двух любых чисел больше третьего.

При построении блок-схемы, чтобы не делать блоки условия слишком большими, вынесем сами условия их в качестве комментариев к блокам.



**Рис.7.5.** Таблица для составления тестового варианта.

Рассмотрим следующие возможные ситуации:

1. Ввод трех вариантов целочисленных значений, чтобы пройти по каждой ветви выбора треугольника (равносторонний, равнобедренный, разносторонний);
2. Ввод вещественных значений;

Так как стороны треугольника могут быть и целого и вещественного типа данных, то программа должна выдавать значение площади треугольника и сообщение о виде треугольника.

1. Ввод значений, равных 0;
2. Ввод отрицательных сторон треугольника;
3. Ввод сторон треугольника, равных символьным значениям;
4. Ввод значений, при которых треугольник не существует.

В случаях (3-6) программа должна выдавать соответствующее сообщение об ошибке, так как треугольника существовать не должно. Если Фактический результат в этих случаях будет выдавать значение площади треугольника или системную ошибку, не обработанную вами, то это ошибочное решение.

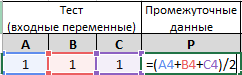
**Примечание:** Старайтесь не вводить множество однотипных данных, так как если данные равнозначны, то, с точки зрения проверки правильности программы, считается, что если программа работает правильно на одном наборе входных данных из этого класса, то она будет правильно работать на любом другом наборе данных из этого же класса.

Входные данные для тестовых вариантов должны быть как можно разнообразнее.

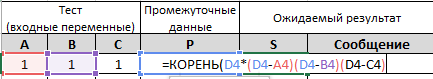
****

**Рис.7.6.** Примерный тестовый вариант.

Для того, чтобы все формулы просчитывались автоматически для всех тестовых вариантов будем пользоваться *Формулы -> Вставить функцию -> Математические*:

****

**Рис.7.7.** Вычисление полупериметра треугольника.

****

**Рис.7.7.** Вычисление площади треугольника.

1. **Задания для самостоятельной работы:**
2. Разработать тестовые варианты для Лабораторная работа №2 - Лабораторная работа №6.
3. **Контрольные вопросы:**
4. Дайте определение понятию «тестирование».
5. Что такое тестовый вариант?
6. Почему при тестировании необходимо желать, чтобы программа выдала ошибку?
7. В чем заключается принцип тестирования программы как белого ящика?
8. В чем заключается принцип тестирования программы как черного ящика?

# Лабораторная работа №8

**Реализации линейных, разветвляющихся и циклических**

**алгоритмов на VBA.**

**Цель работы**

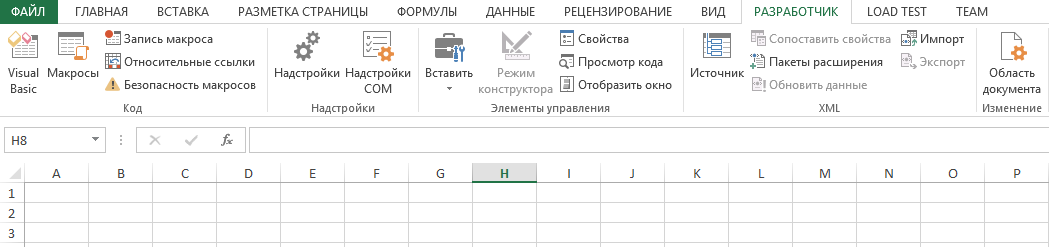
Ознакомление студентов со структурой языка Visual Basic for Applications. Научить реализовывать линейные, разветвляющиеся и циклические алгоритмы в этой среде.

1. **Теоретические сведения:**

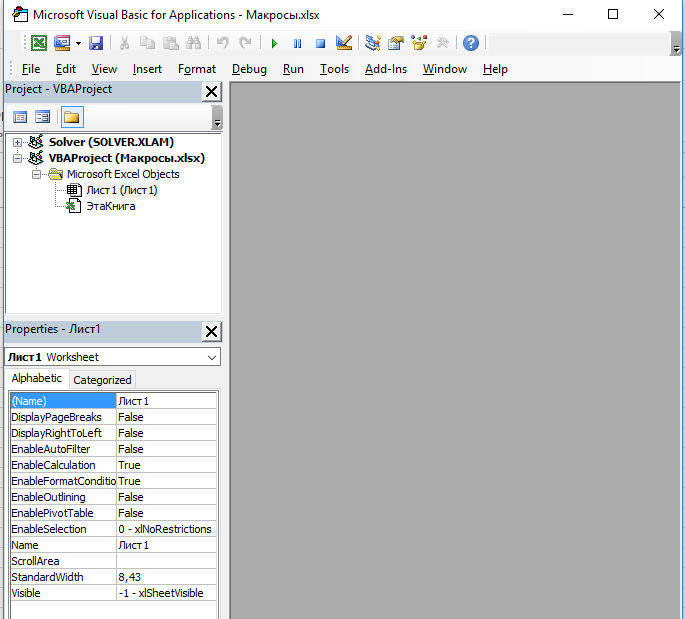
Visual Basic for Applications – это объектно-ориентированный язык программирования, который используется в качестве макроязыка в электронных таблицах и других приложениях. Отличительной особенностью VBA является использование наряду с обычными переменными и константами также и имеющихся объектов приложений MS Office, например, в MS Excel это могут быть рабочие книги, рабочие листы, диапазоны ячеек, диаграммы и т.д.

***Структура редактора VBA.***

Код VBA создается в редакторе Visual Basic. Чтобы запустить этот редактор, необходимо перейти на вкладку Разработчик (рис. 8.1) и в группе Код нажать кнопку Visual Basic: откроется интегрированная среда разработки приложений Visual Basic (рис. 8.2).



**Рис. 8.1**. Вкладка Разработчик.



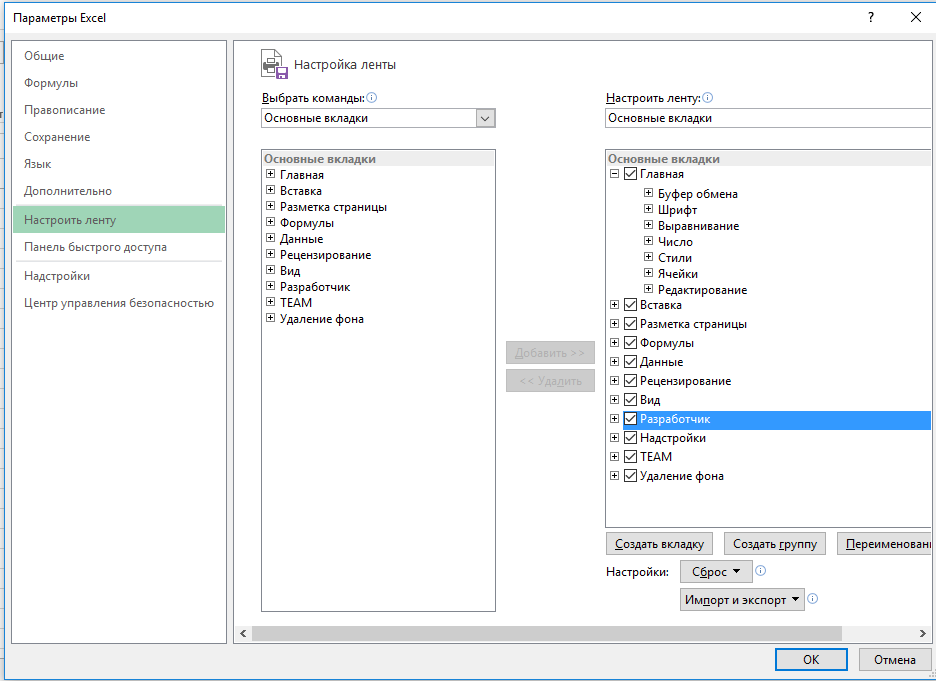
**Рис. 8.2**. Среда разработки приложений Visual Basic.

Если на ленте нет вкладки *Разработчик*, необходимо выполнить следующие действия:

1. Перейти на вкладку *Файл* -> *Параметры* -> *Настройка ленты*-> *Основные вкладки*.

2. Установить флажок *Разработчик* (рис. 8.3) и нажать на кнопку ОК.

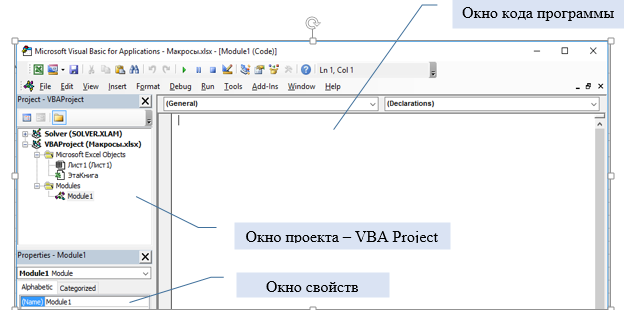
*Примечание:*Для быстрого запуска редактора VBA достаточно нажать комбинацию клавиш <Alt> + <F11>.



**Рис. 8.3**. Окно Параметры Excel.

Интерфейс VBA состоит из следующих основных компонентов: окно проекта, окно свойств, окно редактирования кода, окно форм, меню и панели инструментов. По умолчанию в проект VBA входят рабочий лист и рабочая книга. В окне свойств отображаются все свойства выбранного объекта. В проект можно добавлять дополнительные объекты. Код программы, который не связан с определенным объектом (например, с листом, формой или объектом формы), обычно хранится в модуле (Module).

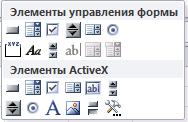
**Модуль** – это объект, в котором хранится код программы с определением переменных, подпрограмм и функций. Для добавления модуля в проект нужно выполнить команду Insert Module. После того как в проект добавлен модуль, в области разработки отобразится код, хранящийся в этом модуле (рис. 8.4).



**Рис. 8.4**. Module.

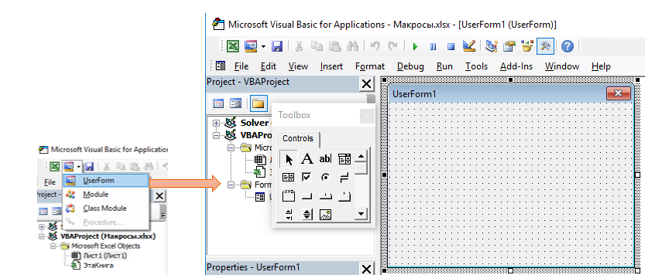
VBA обладает встроенным набором элементов управления.

Элементы управления являются объектами. Поэтому как любые объекты, они обладают *свойствами*, *методами*, *событиями*. Доступные элементы управления отображены на вкладке *Разработчик* в разделе *Элементы управления* (рис. 8.5). Размещение элементов управления на рабочем листе или в форме, как правило, происходит на начальном этапе конструирования приложения. Большинство элементов управления можно располагать как на рабочем листе, так и в форме.



**Рис.8.5.** Элементы управления.

Для создания диалоговых окон в приложениях VBA используются формы. Редактор форм является одним из основных инструментов визуального программирования. Форма в проект добавляется с помощью команды **Insert, UserForm** или нажатием кнопки **InsertUserForm** . В результате на экран выводится незаполненная форма с панелью элементов (Toolbox). Используя панель элементов из незаполненной формы можно сконструировать любое требуемое для приложения диалоговое окно.



**Рис.8.6.** Создание пользовательской формы.

**Константы.**

Программист может создать свои собственные константы с помощью специального оператора Const:

Const *имяКонстанты* = *выражение*

Const *имяКонстанты* As *типДанных* = *выражение,*

где *выражение* — это любое значение или формула, возвращающая значение, которое должно использоваться в качестве константы.

**Сообщения.**

Процедура MsgBox() выводит на экран диалоговое окно, содержащее сообщение, устанавливает режим ожидания нажатия кнопки пользователем. Синтаксис создания окна сообщений:

**MsgBox (Сообщение, Атрибуты, Заголовок)**

*Сообщение* — текст, отображаемый в окне сообщений, является обязательным аргументом. Эта строка должна быть заключена в двойные кавычки.

*Атрибуты* определяют особенности окна, т.е. различные кнопки и значки, отображаемые в нем. Аргумент Атрибуты позволяет управлять следующими параметрами окна сообщения: количеством кнопок в окне, типом кнопок и их размещением в окне, пиктограммой, отображаемой в окне.

Атрибуты = Параметр1 + Параметр2.

|  |  |  |
| --- | --- | --- |
| **Параметр1** | **Набор кнопок** |  |
| 0 | Отображается только кнопка "ОК" |  |
| 1 | Отображаются кнопки "ОК" и "Отмена" |  |
| 2 | Отображаются кнопки "Прервать",  "Повтор", "Пропустить" |  |
|  |
| 3 | Отображаются кнопки "Да", "Нет", "Отмена" |  |
| 4 | Отображаются кнопки "Да", "Нет" |  |
| 5 | Отображаются кнопки "Повтор","Отмена" |  |
|  | |  |
| **Параметр2** | **Вид сообщения** |  |
| 16 | Отображается значок критического сообщения |  |
|  |
|  |
| 32 | Отображается вопросительный знак  (предупреждение) |  |
|  |
|  |
| 48 | Отображается восклицательный знак  (предупреждение) |  |
|  |
|  |
| 64 | Отображается знак информационного сообщения |  |
|  |
|  |
|  |

**Оператор ветвления.**

Для записи разветвляющихся алгоритмов используется оператор If.

Он имеет две формы записи.

1. Однострочная запись **If** условие **Then** [оператор1] [**Else** оператор2]

2. Многострочная запись

If условие1 Then

[ оператор1]

[ElseIf условие2 Then  
 [оператор2] ...

[Else  
 [оператор3]]]

End If

В условии записывается логическое выражение, которое использует операции сравнения, приведенные ниже. Для создания сложных условий используются логические операции ***and*** *(и)* и ***or*** (*или)*

В том случае, если задача состоит из большого количества условий, пользоваться конструкцией If Else не удобно, так как она получится слишком громоздкой и будет очень легко запутаться. В таких случаях можно воспользоваться конструкцией множественного выбора - Select Case, имеющий следующий синтаксис:

Select Case *проверяемоеВыражение*

Case *списокЗначений\_1*

*группаОператоров\_1*

Case *списокЗначений\_2*

*группаОператоров\_2*

Case *списокЗначений\_3*

*группаОператоров\_3*

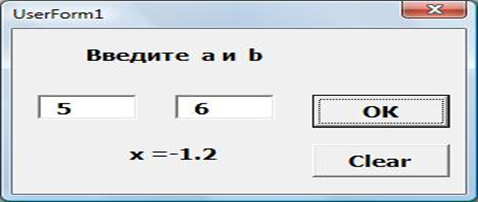
...

Case Else

*группаОператоров\_Else*

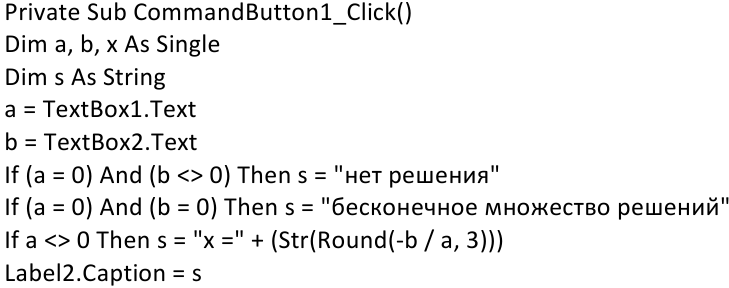
End Select

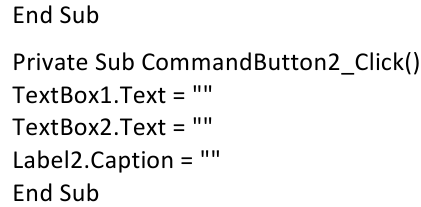
**Пример 1**. Создать программу решения уравнения ax + b = 0. Для ввода значений коэффициентов а и b на окне формы разместить два поля ввода TextBox, результат вывести на надпись (рис. 8.7).



**Рис.8.7**. Вид окна программы.

Код модуля формы:

****

****

**Операторы цикла.**

Циклы, в которых число повторений заранее известно, называются арифметическими циклами. Циклы, в которых известно только условие

завершения цикла, а число повторений цикла заранее неизвестно, называются итерационными циклами.

В VBA есть богатый выбор средств организации циклов, которые можно

разделить на две основные группы:

* 1. *Циклы с перечислением* For…Next
  2. *Циклы с условием* Do … Loop

Цикл For … Next используется для организации арифметических циклов.

Циклы типа Do … Loop используются в тех случаях, когда заранее

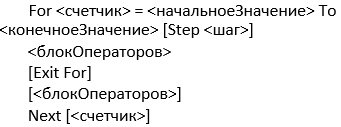
неизвестно сколько раз должно быть повторено выполнение блока операторов, составляющего тело цикла. Такой цикл продолжает свою работу до тех пор,

пока не будет выполнено определенное условие. Такие циклы делятся на:

1. *Циклы с предусловием* Do While…Loop
2. *Циклы с постусловием* Do Until…Loop

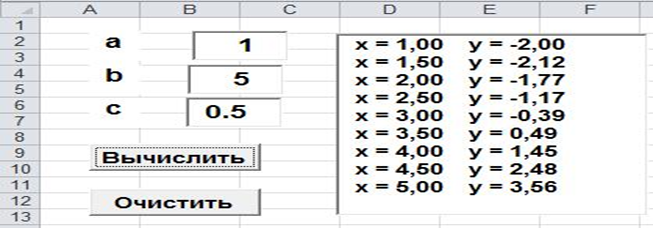
**Цикл For … Next**

Синтаксис:



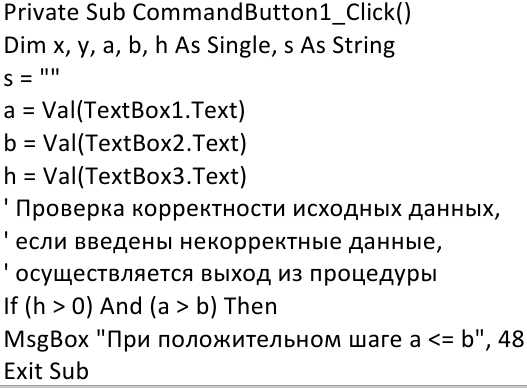
**Пример 2**.Протабулировать функциюy = 3x–4lnx–5на отрезке[a,b] с

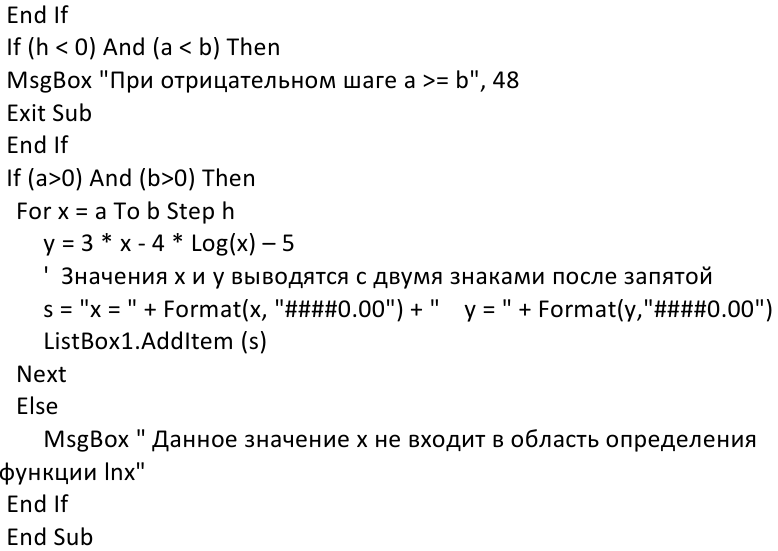
шагом h. Для ввода значений a, b, h на рабочем листе установить три поля ввода TextBox, для вывода значений аргумента и функции – список ListBox, для управления работой программы – две командные кнопки (рис. 8.8).

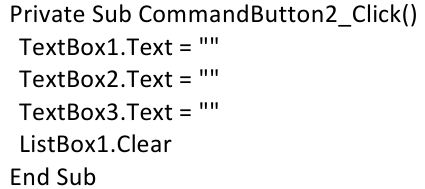


**Рис.8.8.** Вид окна программы.

Код модуля рабочего листа:



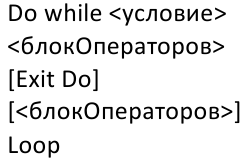




**Оператор цикла с предусловием – Do while.**

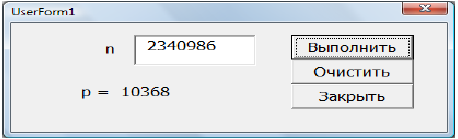
Используется для организации итерационных циклов.

Синтаксис:



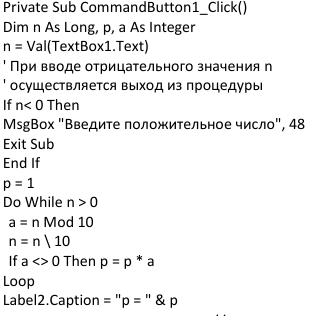
**Пример** **3**. Найти произведение ненулевых цифр натурального числа n.

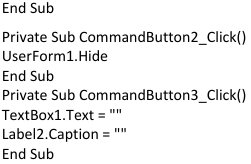
Для ввода значения n на окне формы установить поле ввода TextBox, для вывода значения произведения - надпись (рис.8.9).



**Рис.8.9.** Вид окна программы.

Код модуля формы:





**2. Задания для самостоятельной работы:**

1. Реализовать алгоритмы, разработанные в Лабораторная работа №1 - Лабораторная работа №6.
2. **Контрольные вопросы:**
3. Что такое модуль?
4. Что такое макрос?
5. Каков синтаксис создания окна сообщения?
6. Do While … Loop и Do Until … Loop, отличия?
7. Как быстро вызвать макрос?

# Литература

**Основная литература:**

1. С.А. Орлов. Технологии разработки программного обеспечения. Учебник. - СПб: Питер, 2002.
2. Йан Соммервилл «Инженерия программного обеспечения». Издат. Дом Москва, 2002г.
3. Презентации семинаров по методам тестирования в рамках TEMPUS проекта HEICA.
4. Боэм Б.У. Инженерное проектирование программного обеспечения. М.:Энергоатомиздат,1993.
5. Майерс Г. Искусство тестирования программ. М.: Финансы и статистика, 1982.

**Дополнительная литература:**

1. Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. UML руководство пользователя. М.: ДМ К, 2000.
2. Дин Леффингуэлл, Дон Уидриг. Принципы работы с требованиями к программному обеспечению. М.: Вильяме, 2002.
3. Презентации семинаров по методам тестирования в рамках TEMPUS проекта HEICA. /MagiPocs/Buffer.
4. Леоненков А. В. Самоучитель UML. СПб.: BHV, 2006.
5. Вендров А. М. CASE-технологии. Современные методы и средства проектирования