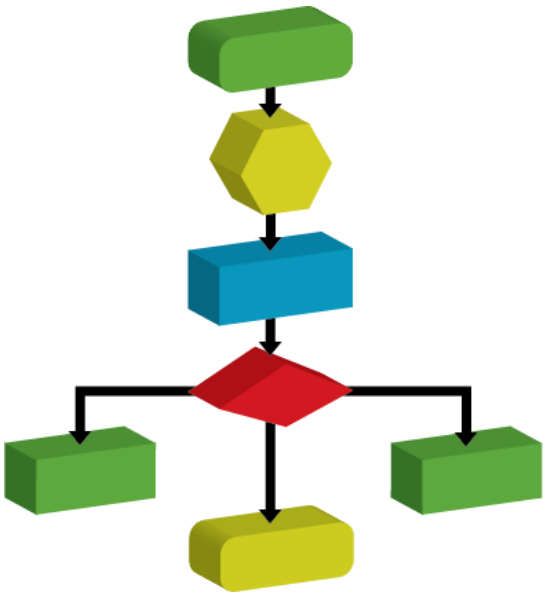
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Брянский государственный аграрный университет»

КАФЕДРА ИНФОРМАТИКИ, ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

УЛЬЯНОВА Н.Д.

ОСНОВНЫЕ ПРИНЦИПЫ АЛГОРИТМИЗАЦИИ

Учебно-методическое пособие
по дисциплине «Алгоритмизация и программирование»



Брянская область,
2020

УДК 681.3.06:004.4 (07)
ББК 32.973.26-018.2
У 51

Ульянова, Н. Д. Основные принципы алгоритмизации: учебно-методическое пособие по дисциплине «Алгоритмизация и программирование» / Н. Д. Ульянова. - Брянск: Изд-во Брянский ГАУ, 2020. - 56 с.

В пособии кратко раскрывается понятие и принципы алгоритмизации, описаны основные структуры создания алгоритмов, приемы построения блок-схем. Представлены задания и последовательность их выполнения при проведении лабораторно-практических работ.

Пособие предназначено для студентов направления подготовки 09.03.03 Прикладная информатика (уровень бакалавриата), а также студентов различных направлений подготовки высших учебных заведений с целью изучения теоретических основ алгоритмизации, основных структур алгоритмов с построением блок-схем и выполнения лабораторно-практических заданий.

Рецензент: к.э.н., доцент кафедры информатики**,** информационных систем и технологий Лысенкова С.Н.

Рекомендовано к изданию решением методической комиссии института энергетики и природопользования, протокол № 2 от 28 октября 2020 г.

© Брянский ГАУ, 2020
© Н.Д. Ульянова, 2020

ВВЕДЕНИЕ

Понятие «алгоритм обработки данных» в компьютерных науках используется для описания метода решения задачи, который в дальнейшем, возможно, реализовать в выбранной среде программирования. Тщательная разработка алгоритма является весьма эффективной частью процесса решения задачи в любой области применения. При разработке алгоритма для реальной задачи значительные усилия должны быть потрачены на осознание степени ее сложности, выяснение ограничений на входные данные, разбиение задачи на менее трудоемкие подзадачи.

Основной целью пособия является формирование навыков построения алгоритмов для решения различных задач в рамках формирования у обучающихся специальных знаний, умений, навыков по дисциплине «Алгоритмизация и программирование». Представленные материалы имеют целью формирование компетенций и освоение обучающимися видов профессиональной деятельности в соответствии с ФГОС ВО по направлению подготовки 09.03.03 Прикладная информатика (уровень бакалавриата).

В предлагаемом пособии рассматриваются понятие и принципы алгоритмизации, описаны основные структуры создания алгоритмов, приемы построения блок-схем. Представлены задания и последовательность их выполнения при проведении лабораторно-практических работ.

Пособие предназначено для студентов направления подготовки 09.03.03 Прикладная информатика (уровень бакалавриата), а также студентов различных направлений подготовки высших учебных заведений с целью изучения теоретических и практических основ построения алгоритмов.

1. ОСНОВНЫЕ ЭТАПЫ КОМПЬЮТЕРНОГО РЕШЕНИЯ ЗАДАЧ

В настоящее время на ЭВМ решают самые разнообразные задачи. В каждом случае ЭВМ выполняет какую-то программу. Некоторые из программ требуют от пользователя специальных знаний и высокой квалификации. Несмотря на бесконечное разнообразие программ, в самом процессе их изготовления можно выделить несколько этапов решения задачи на ЭВМ.

Компьютерное решение задачи включает несколько этапов:

1. *Постановка задачи.* Под постановкой задачи понимают математическую или иную строгую формулировку решаемой задачи. На этом этапе должно быть четко определено:

а) цели создаваемой программы;

б) ограничения, налагаемые на программу,

в) что дано,

г) что требуется найти,

д) состав и организация исходных данных и искомых результатов,

е) время решения поставленной задачи,

ж) объем необходимых ресурсов (например, оперативной памяти),

з) точность достигаемого результата.

Связи между данными описываются с помощью математических формул. Таким образом, на первом этапе задача формализуется, то есть осуществляется сбор информации о задаче; формулировка условия задачи; определение конечных целей решения задачи; определение формы выдачи результатов; описание данных (их типов, диапазонов величин, структуры и т.п.)

2. *Выбор метода решения задачи (проектирование программы, анализ и исследование задачи)*. Для поставленной математической задачи необходимо выбрать метод ее численного решения, сводящий решение задачи к последовательности арифметических и логических операций.

На данном этапе определяют:

- вид данных, с которыми будет работать программа,

- основные части, из которых программа будет состоять,

- характер связей между этими частями.

На данном этапе осуществляется анализ существующих аналогов; анализ технических и программных средств; разработка математической модели; разработка структур данных. Если задача вычислительная, то на этом этапе следует выбрать метод расчета, если разрабатывается компьютерная игра, должен быть определен ее сценарий. Если готовых методов для решения поставленной задачи не существует, то их разрабатывают самостоятельно.

3. *Разработка алгоритма решения задач*. В соответствии с поставленной задачей и методом ее решения описывается последовательность действий по реализации метода. Детализация доводится до той степени, когда перевод действий на алгоритмический язык (кодирование действий программы) станет тривиальным. Разработка алгоритма завершается представлением при помощи одного из способов описания алгоритмов.

На данном этапе осуществляется выбор метода проектирования алгоритма; выбор формы записи алгоритма (блок-схемы, псевдокод и др.); выбор тестов и метода тестирования; проектирование алгоритма. Цель такого представления состоит в том, чтобы еще до этапа программирования убедиться в правильности логики разрабатываемого решения задачи.

4. *Составление программы на языке программирования (кодирование).*

Кодирование алгоритма – запись разработанного алгоритма на алгоритмическом языке. На данном этапе алгоритм описывается на языке программирования, то есть составляется программа.

На данном этапе осуществляется выбор языка программирования; уточнение способов организации данных; запись алгоритма на выбранном языке программирования. Для выполнения данного этапа необходимо знать хотя бы один из многих существующих языков программирования, а лучше знать несколько, чтобы выбрать наиболее подходящий для решаемой задачи.

5. *Ввод программы, ее отладка и тестирование*. На данном этапе проводится контроль правильности работы программы, поиск возможных ошибок и их исправление.

Отладка и тестирование - это два четко различимых и непохожих друг на друга этапа. Тестирование устанавливает факт наличия ошибок, а отладка выясняет ее причину.

Тестирование - это испытание, проверка правильности работы программы в целом, либо её составных частей.

Отладка программы - это процесс поиска и устранения ошибок в программе, производимый по результатам её прогона на компьютере.

На данном этапе осуществляется синтаксическая отладка (нарушение грамматики алгоритмического языка); отладка семантики и логической структуры; тестовые расчеты и анализ результатов тестирования; совершенствование программы.

6. *Решение задачи на ЭВМ и анализ результатов.* На данном этапе компьютер выполняет все предусмотренные программой вычисления, выдает результаты на экран монитора или печатающее устройство, а затем пользователь интерпретирует полученные результаты в соответствии с поставленной целью.

На данном этапе осуществляется анализ результатов решения задачи и уточнение в случае необходимости с повторным выполнением этапов 2 - 5.

7. *Сопровождение программы*. Сопровождение программы - это работы, связанные с обслуживанием программы в процессе ее эксплуатации.

На данном этапе проводится доработка программы для решения конкретных задач; а также составление документации к решенной задаче, к математической модели, к алгоритму, к программе, к набору тестов, к использованию.

Программа живет, приобретает новые функции, совершенствует старые, избавляется от последних ошибок и, наконец, умирает, уступив натиску более новых и совершенных программ.

Рассмотренную последовательность этапов называют *технологической цепочкой компьютерного решения задачи*.

2. АЛГОРИТМИЗАЦИЯ. ПОНЯТИЕ АЛГОРИТМА. ОСНОВНЫЕ СВОЙСТВА АЛГОРИТМОВ

Любая задача, решаемая на компьютере, преследует какую-нибудь цель. Для достижения этой цели необходимо как следует сформулировать желаемый результат, потом продумать четкий план его достижения, тогда на компьютере цель будет достигнута. Этот процесс формирования плана последовательности действий называется алгоритмизацией.

Алгоритмизация - сведение задачи к последовательности этапов, выполняемых друг за другом так, что результаты предыдущих этапов используются при выполнении следующих.

Это согласуется с возможностями компьютера выполнять действия последовательно одно за другим. Результатом выполнения этапа алгоритмизации является *алгоритм решения задачи.*

Слово «алгоритм» происходит от имени выдающегося персидского математика одного из крупнейших среднеазиатских учёных IX века, математика, астронома, географа и историка Мухаммеда аль-Хорезми. Им были предложены приемы выполнения арифметических вычислений с многозначными числами. Позже в Европе эти приемы назвали алгоритмами от «Algorithmi» - латинского написания имени ученого.

В начале XX века алгоритмы стали объектом изучения математиков, появились различные математические уточнения понятия «алгоритм» и возникла целая отрасль математики – теория алгоритмов. Результаты, полученные теорией алгоритмов, служат теоретическим фундаментом всей компьютерной технологии, но в повседневной программистской практике не используются.

В наше время термин «алгоритм» понимается шире, не ограничиваясь только арифметическими вычислениями. Он применяется не только в информатике, но и в быту. Под алгоритмом понимают описание какой-либо последовательности действий для достижения заданной цели. В этом смысле алгоритмом можно назвать инструкцию по использованию кухонного комбайна, кулинарный рецепт.

В информатике существует несколько определений понятия алгоритма:

1) *алгоритм* – конечная последовательность общепонятных предписаний, формальное, не требующее проявления человеческой изобретательности, исполнение которых позволяет за конечное время получить решение некоторой задачи;

2) *алгоритм* – система правил, четко описывающая последовательность действий, которые необходимо выполнять для решения задачи;

3) *алгоритм* – понятное и точное предписание исполнителю выполнить конечную последовательность действий, приводящую от исходных данных к искомому результату.

Данные определения являются истинными и заключают в себе один и тот же смысл.

Исполнитель алгоритма - это некоторая абстрактная или реальная (техническая, биологическая или биотехническая) система, способная выполнить действия, предписываемые алгоритмом. В информатике универсальным исполнителем алгоритмов является *компьютер*.

Алгоритмы бывают численными и логическими.

Численный алгоритм - алгоритм, в соответствии с которым решение поставленной задачи сводится к арифметическим действиям.

Логический алгоритм - алгоритм, в соответствии с которым решение поставленной задачи сводится к логическим действиям.

Например, алгоритмы поиска минимального числа, поиска пути в лабиринте.

Основные свойства алгоритма.

1. *Понятность* - содержание допустимого набора команд, понятного конкретному исполнителю.

Алгоритм, составленный для конкретного исполнителя, должен включать только те команды, которые входят в его систему команд. Алгоритм не должен быть рассчитан на принятие каких-либо самостоятельных решений исполнителем. Это облегчает проверку и модификацию алгоритма при необходимости.

2. *Дискретность* (прерывность, раздельность) - разделение выполнения решения задачи на отдельные операции. Алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов (этапов).

При составлении алгоритма процесс решения задачи должен быть разбит на последовательность отдельных шагов. Таким образом, формируется упорядоченная совокупность отдельных друг от друга команд. Образующаяся структура алгоритма оказывается прерывной (дискретной): только выполнив одну команду, исполнитель может приступить к выполнению следующей.

3. *Определенность (детерминированность)* - каждая команда алгоритма должна быть точной, четкой, однозначной и не оставлять места для произвольного толкования действия исполнителя.

Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.

4. *Результативность (конечность)* – завершение работы алгоритма за конечное число шагов, которое должно привести к выдаче результатов или сообщению о невозможности решения задачи.

При этом количество шагов может быть заранее не известным и различным для разных исходных данных.

5. *Массовость (универсальность)* – пригодность алгоритма для решения задач одного и того же типа, но отличающихся различными исходными данными. Алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными.

Свойство массовости не является необходимым свойством, оно скорее определяет качество алгоритма. Свойства определенности, конечности и понятности являются необходимыми (иначе это не алгоритм).

3. ГРАФИЧЕСКИЙ СПОСОБ ОПИСАНИЯ АЛГОРИТМОВ

Для алгоритма строго не определяется форма его представления. Существуют различные системы правил и формальных обозначений для задания алгоритмов. Такие системы правил называются *языками описаний*.

К основным формам представления алгоритмов относят:

1) *словесная* - записи 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).

Все способы представления алгоритмов можно считать взаимодополняющими друг друга. На этапе проектирования алгоритмов наилучшим способом является графическое представление, а на этапах проверки и применения алгоритма – запись в виде программы.

На практике наиболее распространены алгоритмы в графической форме. Они считаются более компактными и наглядными, имеют ряд преимуществ благодаря визуальности и явному отображению процесса решения задачи.

Графический – это способ представления алгоритма с помощью геометрических фигур, называемых функциональными блоками, соединенных линиями.

Последовательность блоков и соединительных линий образуют *блок-схему*.

Блок-схема - наглядное изображение алгоритма, когда отдельные действия (этапы алгоритма) изображаются при помощи различных геометрических фигур (блоков), а связи между этапами (последовательность выполнения этапов) указываются при помощи стрелок, соединяющих эти фигуры.

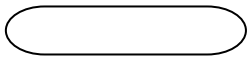
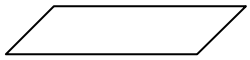
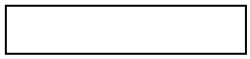
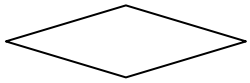
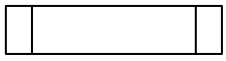
В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т.п.) соответствует геометрическая фигура, представленная в виде блочного символа. Блочные символы соединяются линиями переходов, определяющими очередность выполнения действий.

Любая блок-схема имеет одно начало и один конец. Блоки схемы располагаются сверху вниз. Каждый блок соответствует определенному типу действий и представляется в виде геометрической фигуры. Внутри каждого блока указывается поясняющая информация, которая характеризует действия, выполняемые этим блоком.

Формы блоков и правила составления схем алгоритмов установлены гостом. На территории Российской Федерации действует единая система программной документации (ЕСПД), частью которой является Государственный стандарт - ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем». Данный ГОСТ практически полностью соответствует международному стандарту ISO 5807:1985. В таблице приведены наиболее часто употребляемые символы.

Таблица

Перечень блоков, используемых при графическом представлении алгоритмов

Название символа	**Обозначение**	**Пояснение**
Терминатор начала и конца работы		Начало или конец алгоритма, вход или выход в подпрограммах
Ввод-вывод		Ввод данных и вывод результатов
Процесс		Вычислительное действие или последовательность действий
Решение		Проверка условий и выбор направления выполнения алгоритма
Подготовка (Модификация)		Выполнение действия, изменяющего пункты (например, заголовок цикла)
Предопределенный процесс		Вычисления по подпрограмме, стандартной подпрограмме
Соединитель		Указание связи между потоками информации в пределах одного листа
Межстраничный соединитель		Указание связи между информацией на разных листах
Комментарий		Пояснительные записи в целях объяснения или примечаний

Линии соединения (переходов) отдельных блоков показывают направление процесса обработки в схеме, определяют очередность выполнения действий. Они должны проводиться по вертикали и горизонтали под углом 90^0. Направление потока слева направо и сверху вниз считается стандартным, поэтому стрелки можно не ставить.

Количество входящих линий не ограничено, выходящая линия из блока должна быть одна, за исключением логического блока.

В схемах следует избегать пересечения линий. Пересекающиеся линии не имеют логической связи между собой, поэтому изменения направления в точках пересечения не допускаются. Если две или более входящих линии объединяются в одну исходящую линию, то место объединения линий смещается.

В настоящее время существуют технологии разработки программ без использования блок-схем. Но на начальном этапе программирования использование схем при разработке алгоритма целесообразно, так как является основой структурного подхода.

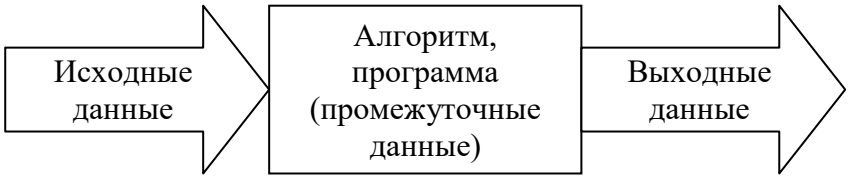
4. ДАННЫЕ И ИХ ТИПЫ

Алгоритм, реализующий решение задачи, всегда работает с данными.

Данные – это любая информация, представленная в формализованном виде и пригодная для обработки алгоритмом.

*Смысловое (семантическое) разбиение данных производится во время постановки задачи и разработки алгоритма ее решения.

По отношению к алгоритму данные делятся на исходные, промежуточные и выходные.



Исходные данные (начальные) - данные, известные перед выполнением алгоритма.

Промежуточные данные - данные, используемые в процессе выполнения программы.

Выходные данные - результат решения задачи, конечные данные.

Данные делятся на переменные и константы.

Переменные – это данные, значения которых могут изменяться в процессе выполнения алгоритма.

Константы – это данные, значения которых не меняются в процессе выполнения алгоритма.

Для программной обработки в ЭВМ данные представляются в виде величин и их совокупностей.

Величина - это элемент данных с точки зрения их семантического (смыслового) содержания или обработки.

Любая величина имеет *3 основные свойства*:

1) *имя* - обозначение и место величины в памяти,

задается идентификатором, представляющим собой последовательность букв и цифр, начинающихся с буквы;

2) *значение* - динамическая характеристика, которая может меняться многократно в ходе исполнения алгоритма;

Во время выполнения алгоритма в каждый конкретный момент величина имеет какое-то значение или не определена.

3) *тип данных* – характеристика данных, которая задает множество допустимых значений и определяет множество операций, которые можно к этим данным применить.

Любой тип данных должен быть охарактеризован областью значений и допустимыми операциями над этим типом данных.

Типы данных делят на 2 группы:

1) *Простые (скалярные) типы* – содержат одно единственное значение.

   К ним относятся:

- *целый тип* – определяет подмножество допустимых значений из множества целых чисел (например: 23, -12);
- *вещественный тип* - определяет подмножество допустимых значений из множества целых и дробных чисел в некотором диапазоне (например: 2,5; -0,01; $3{,}6 \cdot 10^9$);
- *логический тип* – переменная принимает только два значения: истина (true) и ложь (false);
- *символьный тип* – любые символы компьютерного алфавита (например: ‘a’, ‘5’, ‘+’).

2) *Структурированные типы* - описывают наборы однотипных или разнотипных данных (т.е. содержат несколько значений), с которыми алгоритм должен работать как с одной именованной переменной. К ним относятся: массивы, строки, множества и т.д.

5. ЛОГИЧЕСКИЕ ОСНОВЫ АЛГОРИТМИЗАЦИИ

5.1. ОСНОВЫ ЛОГИКИ

Составляющей алгоритмов являются логические условия, вычисление значений которых происходит в соответствии с аксиомами алгебры логики.

Логика – это наука о формах и способах мышления. Основными формами мышления являются понятие, высказывание и умозаключение.

Первые учения о логике появились в Древнем Востоке. Основоположником формальной логики является Аристотель (IV до н.э.), а основы математической логики заложил английский математик Джордж Буль. Основу математической логики составляет алгебра высказываний.

Алгебра логики - раздел математики, который оперирует логическими высказываниями. Алгебра логики используется при построении основных узлов ЭВМ – шифратора, дешифратора, сумматора.

Логическое высказывание - любое предложение в повествовательной форме, о котором можно однозначно сказать, истинно оно или ложно.

Примеры логических высказываний:

- «Москва - столица России» (высказывание истинно).
- «После зимы наступает осень» (высказывание ложно).

Простое высказывание - логическое высказывание, состоящее из одного утверждения.

Логическая переменная - простое высказывание, содержащее только одну мысль. Обозначается прописными буквами латинского алфавита (например, A, B, C) и могут принимать лишь два значения ИСТИНА (1) и ЛОЖЬ (0).

Над высказываниями можно производить определенные логические операции, в результате которых получаются новые составные (сложные) высказывания.

Сложное высказывание - логическое высказывание, состоящее из нескольких утверждений, объединенных с помощью «связок»: союзов «и», «или (либо)», частицы «не», связки «если, то» и др.

Примеры сложных высказываний:

1. «Иван сдает экзамен по физике **и** информатике».

*Высказывание содержит два утверждения, объединенных «и»:

- Утверждение1: «Иван сдает экзамен по физике».
- Утверждение2: «Иван сдает экзамен по информатике».

2. «Игорь решил записаться в секцию по волейболу **или** баскетболу».

3. **«Если** Илья будет много готовиться самостоятельно **и** будет заниматься с репетитором, **то** он поступит в ВУЗ».

Логические операции – логические действия.

Логические операции – «связки»: союзы и частицы естественного языка, образующие из простых высказываний сложные, представленные в формальном виде.

Логическое выражение - простое или сложное логическое высказывание, представленное в формальном виде.

Примеры логических выражений:

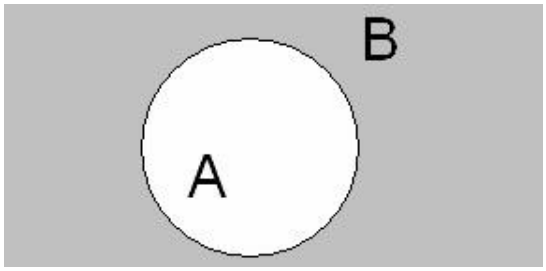
- простое: A,
- сложное: AVB→C,

где A, B, C - утверждения; V, → - логические операции.

Обозначения для логических связок (операций):

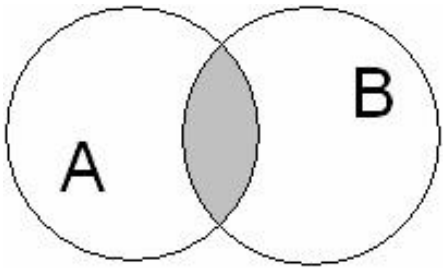
a) *отрицание* (инверсия, логическое НЕ) обозначается ¬ (например, ¬A) или $\overline{A}$;

В физическом смысле логическое отрицание - это сочетание двух областей.



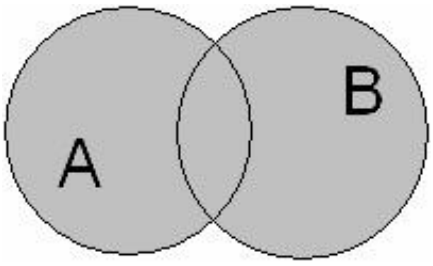
b) *конъюнкция* (логическое умножение, логическое И) обозначается /\ (например, A /\ B) либо & (например, A & B);

В физическом смысле логическое умножение - это пересечение двух областей.



c) *дизъюнкция* (логическое сложение, логическое ИЛИ) обозначается \/ (например, A \/ B) либо | (например, A | B);

В физическом смысле логическое сложение - это объединение двух областей.



d) *следование* (импликация) обозначается → (например, A → B);

e) *тождество* (эквиваленция) обозначается ≡ (например, A ≡ B) или ↔. Выражение A ≡ B истинно тогда и только тогда, когда значения A и B совпадают (либо они оба истинны, либо они оба ложны);

f) символ 1 используется для обозначения истины (истинного высказывания); символ 0 – для обозначения лжи (ложного высказывания).

Приоритеты логических операций: инверсия (отрицание), конъюнкция (логическое умножение), дизъюнкция (логическое сложение), импликация (следование), тождество.

Таким образом, ¬A /\ B \/ C /\ D означает то же, что и ((¬A) /\ B) \/ (C /\ D). Возможна запись A /\ B /\ C вместо (A /\ B) /\ C. То же относится и к дизъюнкции: возможна запись A \/ B \/ C вместо (A \/ B) \/ C.

Задача: Определить порядок выполнения функции AV¬BΛC→D↔E.

Решение:

¬B (¬B)ΛC AV((¬B)ΛC) (AV((¬B)ΛC))→D ((AV((¬B)ΛC))→D)↔E

Два логических выражения, содержащих переменные, называются *равносильными (эквивалентными*), если значения этих выражений совпадают при любых значениях переменных. Так, выражения A → B и (¬A) \/ B неравносильны (значения выражений разные, например, при A = 1, B = 0).

5.2. ТАБЛИЦЫ ИСТИННОСТИ ЛОГИЧЕСКИХ ВЫРАЖЕНИЙ

Таблица истинности выражения определяет его значения при всех возможных комбинациях исходных данных.

Таблица истинности - таблица, которая используется для описания логических функций, в частности отдельных логических операций.

1. *Конъюнкция (логическое умножение - И)* - результат будет истинным тогда и только тогда, когда оба исходных высказывания истинны.

A	B	A ^ B
0	0	0
0	1	0
1	0	0
1	1	1

2. *Дизъюнкция (логическое сложение - ИЛИ)* - результат будет истинным тогда, когда истинно хотя бы одно из высказываний.

A	B	A ⩛ B
0	0	0
0	1	1
1	0	1
1	1	1

3. *Эквивалентность* - результат будет истинным тогда, когда оба исходных высказывания либо истинны, либо ложны.

A	B	A ~ B
0	0	1
0	1	0
1	0	0
1	1	1

4. *Инверсия (логическое отрицание – НЕ*) - результат будет истинным, если исходное высказывание ложно, и наоборот, ложным - если исходное высказывание истинно.

A	$\bar{A}$
0	1
1	0

5. *Импликация (логическое следование)* - результат будет ложным только тогда, когда из истинного высказывания следует ложное.

A	B	A → B
0	0	1
0	1	1
1	0	0
1	1	1

Задача 3. Составить таблицу истинности для функции f (ab) = B∨(B&A)

Решение:

A	B	B&A	BV(B&A)
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	1

Диаграмма Эйлера-Венна - наглядное средство для работы со множествами. Множества изображаются в виде кругов (если используется 2-3 множества) и эллипсов (если используется 4 множества), помещенных в прямоугольник (универсум). Диаграммы Эйлера-Венна используются для визуального представления логических операций.

Пример.

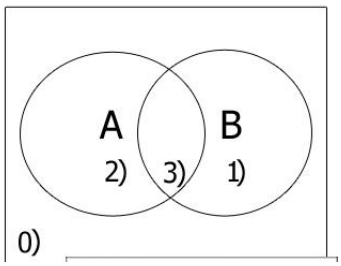
Пусть есть следующие множества чисел:

A={1,2,3,4}

B={3,4,5,6}

Универсум U={0,1,2,3,4,5,6}

Диаграммы Эйлера-Венна для двух множеств A и B:



Определим области, и числа которые им принадлежат:

A	B	Обозначение области	Числа
0	0	0)	0
0	1	1)	5,6
1	0	2)	1,2
1	1	3)	3,4

6. ОСНОВНЫЕ СТРУКТУРЫ АЛГОРИТМОВ

Основные структуры алгоритмов используются при структурном подходе к разработке алгоритмов и программ, предполагающем использование нескольких основных структур, комбинация которых дает все многообразие алгоритмов и программ.

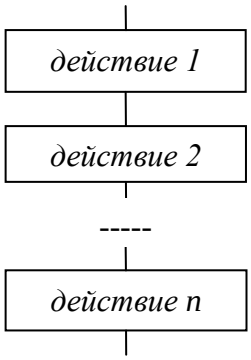
Основные структуры алгоритмов (ОСА) – это определенный набор блоков и стандартных способов их соединения для выполнения типичных последовательностей действий.

Все алгоритмические процессы обработки информации можно свести к трем типам: линейный, ветвящийся (разветвляющийся), циклический. Алгоритм любой сложности можно разработать, используя данные базовые структуры.

К основным алгоритмическим структурам относятся линейные, разветвляющиеся и циклические структуры.

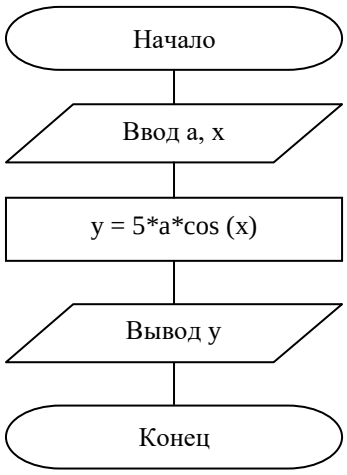
6.1. ЛИНЕЙНЫЕ И РАЗВЕТВЛЯЮЩИЕСЯ СТРУКТУРЫ АЛГОРИТМОВ

Линейный алгоритм – процесс, в котором этапы вычислений производятся последовательно, одно за другим, и каждый этап алгоритма выполняется только один раз.



Типичный пример из жизни – рецепт пирога. На схеме линейный процесс представляется последовательностью блоков, размещаемых сверху вниз.

Пример 1. Вычислить y= 5*a*cos (x)

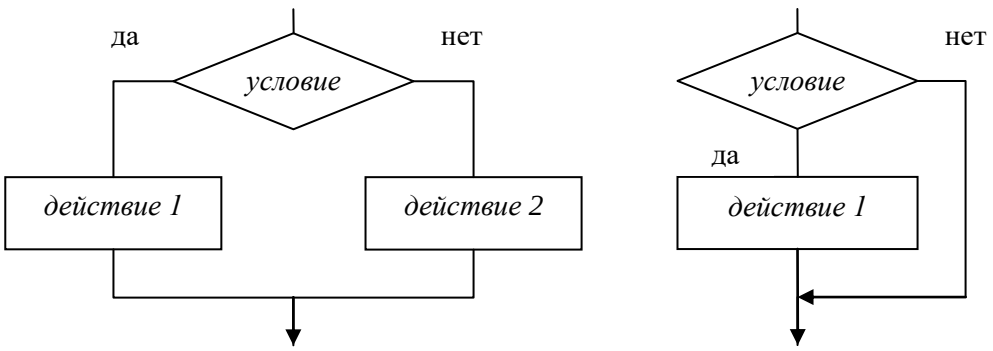


Придавая переменным a и x различные значения, можно использовать полученный алгоритм для вычисления различных y.

При составлении алгоритмов часто возникает необходимость направлять процесс обработки информации по одному из двух возможных путей, то есть выполнять те или иные действия в зависимости от некоторого условия. Так как результат выполнения условия заранее неизвестен, то его проверка должна осуществляться в ходе алгоритма. Процессы такого типа называются ветвящимися.

Ветвящийся (разветвляющийся) алгоритм – процесс, в котором реализация алгоритма происходит по одному из нескольких возможных направлений в зависимости от исходных условий и промежуточных результатов.

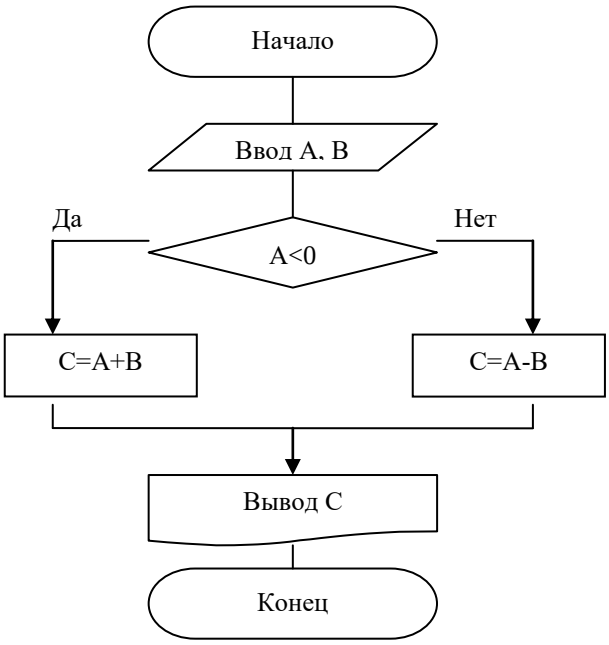
В данных алгоритмах вычислительный процесс выполняется только по одной ветви, а выполнение остальных исключается. В качестве условия в ветвлении используется логическое выражение. Таким образом, алгоритм ветвления состоит из условия и двух последовательных действий.



или

Пример из жизни – правило перехода улицы по светофору…

Пример 2. Вычислить величину $C = \begin{cases} A + B, если A < 0 \\ A - B, если A \geq 0 \end{cases}$



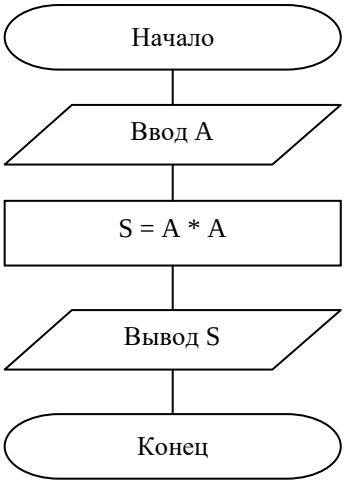
Рассмотренный вариант ветвления называется *полным* ветвлением. Если же на ветви «Нет» отсутствует последовательность команд, то такое ветвление называется *неполным*. Если на ветвях развилки в свою очередь находится ветвление, то говорят, что алгоритм имеет структуру *вложенных* ветвлений.

Лабораторная работа 1. Линейные алгоритмы.
ПП Конструктор алгоритмов

Упражнение 1. Линейные алгоритмы

Задача 1. Разработать алгоритм: задана сторона квадрата A, найти его площадь S

Решение:



- Начало алгоритма
- Ввод значений переменных
- Вычисление площади квадрата
- Вывод результата
- Завершение алгоритма

Задача 2. Разработать алгоритм: даны числа a и b, найти произведение этих чисел.

Задача 3. Разработать алгоритм: даны числа a и b, поменять их местами используя, дополнительную переменную c.

Задача 4. Вы получили наследство 1 000 000$ и хотите красиво пожить. После долгих раздумий вы решаете, что будете жить на … $ в месяц. Разработать алгоритм, который вычисляет, на сколько лет вам хватит наследства.

Упражнение 2. Построение блок схем с помощью программы Конструктор алгоритмов

Задача 1: решить Задачу 1 Упражнения 1 (Разработать алгоритм: задана сторона квадрата A, найти его площадь S) с использованием программы Конструктор алгоритмов.

Порядок работы:

1. Запустить программу Конструктор алгоритмов – **файл 9_14.exe** в папке **Конструктор алгоритмов**.

2. В пункте меню **Блок-схема** выбрать пункт **Новая блок-схема.**

Выбрать пункт меню **Разработка**, тем самым вы переключите программу в режим разработки и редактирования алгоритма.

3. На **Панели блоков** выбрать пункт **Начало схемы алгоритма** и щелкнуть на рабочем поле - появятся блоки «Начало схемы алгоритма» и «Конец схемы алгоритма»

4. Добавить **Блок описания переменных, затем активировать** блок двойным щелчком и нажать кнопку **Редактировать**. В появившимся окне ввести имена переменных (рис. 1).

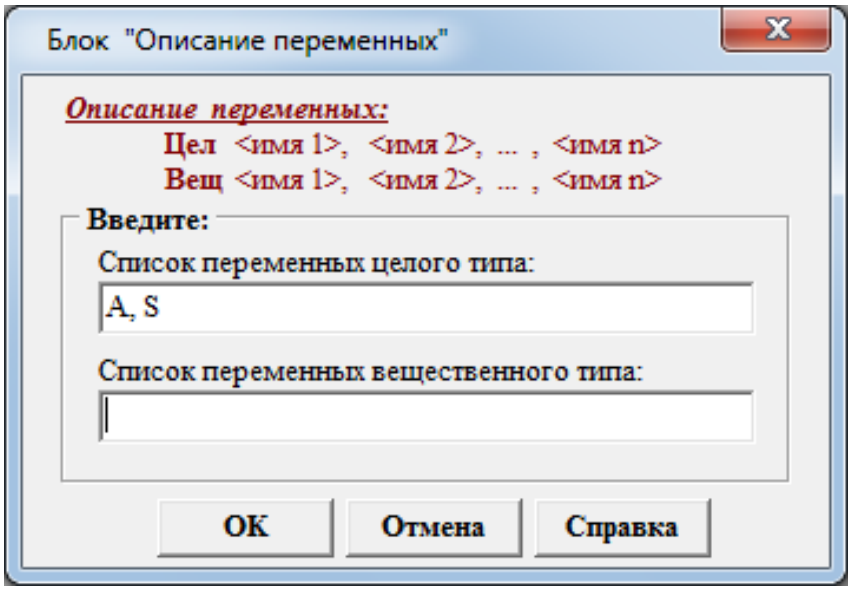


Рис. 1. Окно «Описание переменных»

5. Ввести исходные данные: выбрать блок **Ввод данных** и вставьте его в блок-схему. Активировать блок **Ввода данных** и нажмите **Редактировать**. Ввести переменную А.

6. Вставить блок **Присваивание**. В редактировании блока записать формулу площади квадрата (рис. 2).

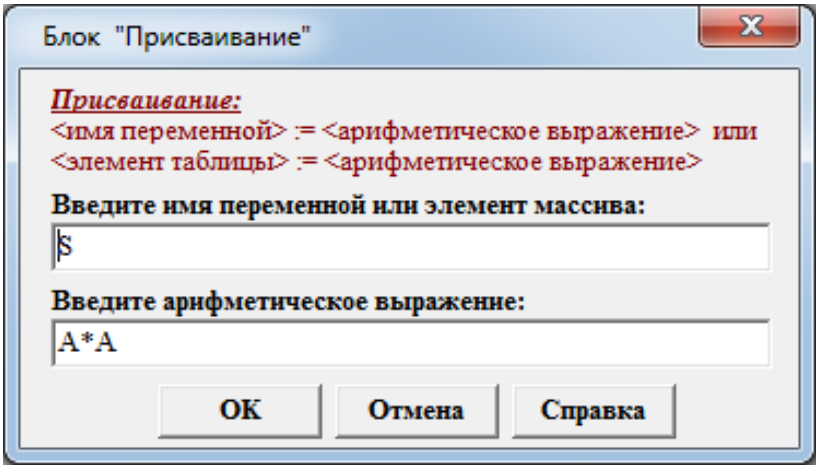


Рис. 2. Окно «Присваивание»

7. Добавить блок **Вывод данных.** Отредактировать блок **Вывода данных**.

8. Проверить алгоритм: перейти в режим **Отладки,** нажать кнопку запуска. Пройти алгоритм пошагово для значения А=5.

9. Выполнить алгоритм целиком для собственного исходного значения А.

10. Сохранить блок-схему в свою папку.

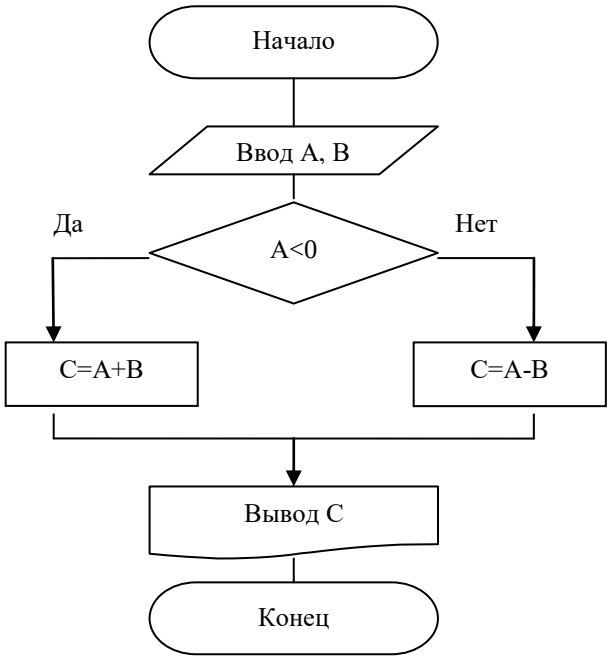
Лабораторная работа №2. Ветвящиеся алгоритмы

Упражнение 1. Ветвящиеся алгоритмы

Задача 1. Разработать алгоритм, который вычисляет величину

$$C = \begin{cases} A + B, если A < 0 \\ A - B, если A \geq 0 \end{cases}$$

Решение:



Задача 2. Разработать алгоритм: даны два значения целого типа x, y; переменной z присвоить максимальное из двух значений.

Задание: решить Задачу 2 Упражнения 1 с использованием программы Конструктор алгоритмов, используя блок Решение (Ветвление).

Задача 3. Разработать алгоритм: заданы целые значения x и y, определить z = max (x, y) и b = min (x,y).

Упражнение 2. Ветвящиеся алгоритмы с логическим типом данных

Логический тип данных имеет всего два значения **True** (истина), **False** (ложь) и является упорядоченным типом **True > False**.

Существуют следующие логические операции:

1. *Операции сравнения:*

   > - больше;

   < - меньше;

= - равно;

<> - не равно;

>= - больше либо равно;

<= - меньше либо равно.

2. ***or*** *(или)* - *логическое сложение (дизъюнкция).*
3. ***and (и)*** - *логическое умножение (конъюнкция).*
4. ***not*** *(не)* - *логическое отрицание.*

Задача 1. Разработать алгоритм: даны два значения целого типа x, y. Если значения x, y положительны, то заменить их нулем, в противном случае оставить их без изменений.

Задача 2. Заданы три стороны треугольника. Определить существует ли такой треугольник, в соответствии с этим вывести «С**уществует», «Не существует».**

Упражнение 3. Алгоритмизация сложных условий

Задача 1. Разработать алгоритм, который вычисляет величину

$$Y = \begin{cases} x^2, если x > 1 \\ x, если 0 \le x \le 1 \\ -2x + 1, если x < 0 \end{cases}$$

Задача 2. Разработать алгоритм: заданы переменные x, y целого типа, вычислить величину

$$A = \frac{\max(x, y) + \min(x, y)}{\max(5, xy) - min(\sqrt{x}, 8 + \sqrt{y})}$$

Задача 3. Из трёх монет одинакового достоинства одна фальшивая (более лёгкая). Как её найти с помощью одного взвешивания на чашечных весах без гирь?

6.2. ЦИКЛИЧЕСКИЕ СТРУКТУРЫ АЛГОРИТМОВ

При реализации алгоритмов многих задач наблюдается многократное повторение отдельных этапов их вычислительного процесса. Такие многократно повторяемые части алгоритмов называются циклами.

Цикл – это команда исполнителю многократно повторить указанную последовательность команд.

Циклический алгоритм – процесс, содержащий многократно повторяемые этапы, то есть содержащий один или несколько циклов.

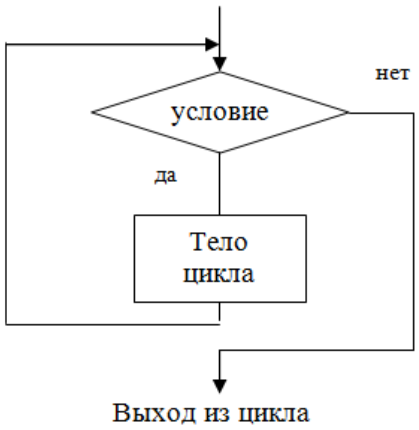
Циклический процесс позволяет значительно сократить длину записываемого алгоритма, а, следовательно, и создаваемой программы, так как не требуется повторять запись одних и тех же операций несколько раз.

Пример циклического алгоритма из жизни – покраска досок.

Циклический процесс включает: блок проверки условия P и блок S, называемый телом цикла. Выделяются следующие виды циклов:

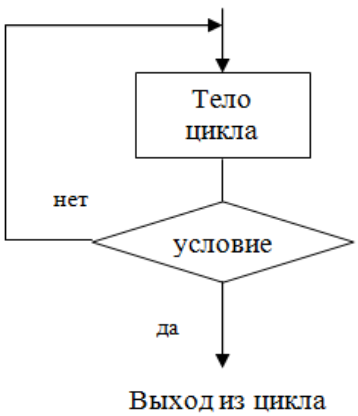
1. *цикл с неизвестным заранее числом повторений*

1.1. *цикл-пока* – когда тело цикла расположено после проверки условия

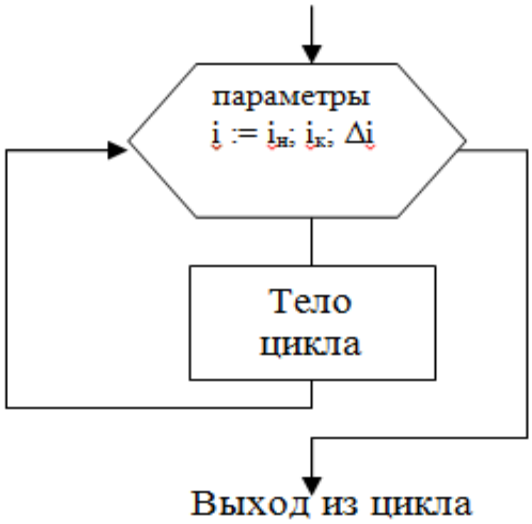


Может случиться, что при определенных условиях блок тела цикла не выполнится ни разу.

1.2. *цикл-до* – когда тело цикла выполняется, по крайней мере, один раз и будет выполняться до тех пор, пока не станет истинным условие



2. *цикл с заданным числом повторений* – для его управления используется специальная переменная – *счетчик*, перед началом выполнения цикла должны быть определены *начальное* и *конечное* значение счетчика, *шаг* его изменения.



Переменная счетчик получает свое начальное значение, которое изменяется на величину шага каждый раз при очередном выполнении цикла.

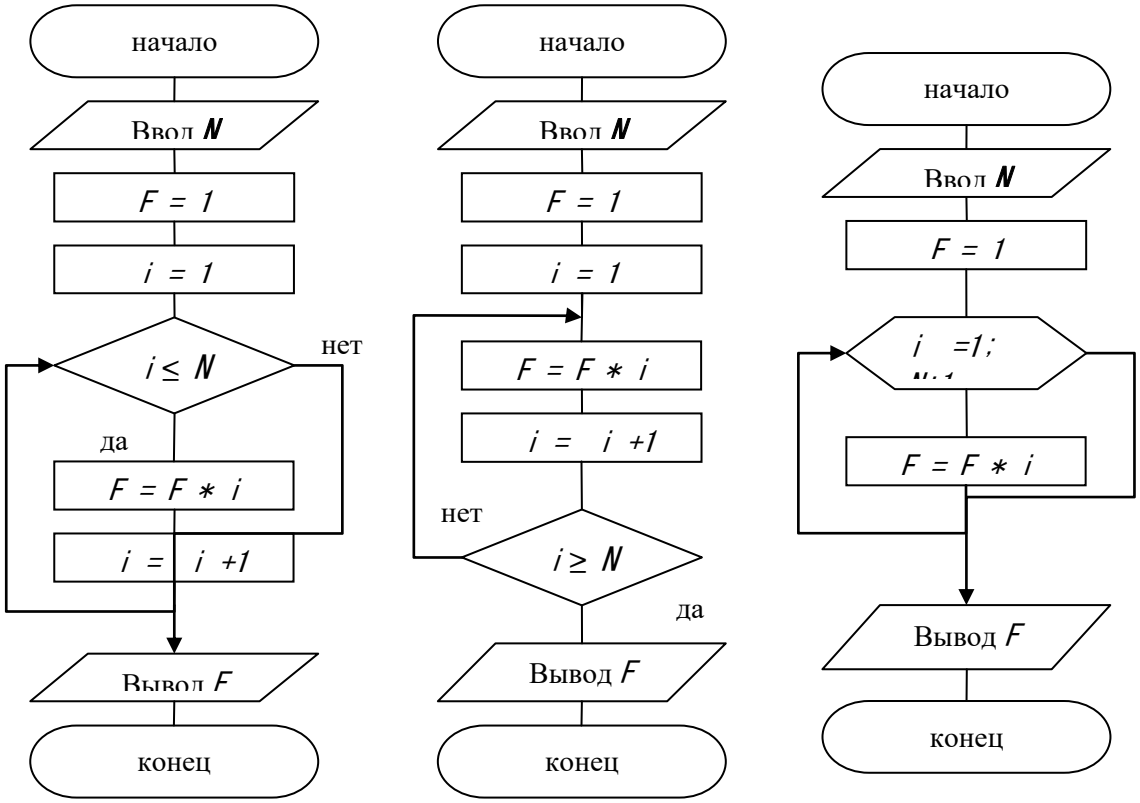
Цикл повторяется до тех пор, пока счетчик не достигнет своего конечного значения.

При вычислении конечной суммы в циклическом алгоритме предварительно необходимо начальную сумму приравнять нулю (S = 0), а при вычислении конечного произведения – начальное произведение приравнять единице (P = 1).

Пример 3. Разработать блок-схему алгоритма вычисления факториала (F) натурального числа N.

Факториал числа (!) – это произведение всех натуральных чисел от 1 до N.

N! = 1*2*3*…* N, (0! =1)



а) цикл с предварительным условием

б) цикл с последующим условием

в) цикл с параметром

Если задача сложна, то ее разбивают на меньшие и простые. Для каждой – отдельный алгоритм. Если подзадачи останутся громоздкими, то они тоже разбиваются на алгоритмы. Процесс продолжается до тех пор, пока задачи не станут простыми. Процесс называется проектированием сверху вниз. При разработке алгоритма допустимо, но менее приемлемо проектирование снизу вверх, когда уже разработаны алгоритмы решения задач самого нижнего уровня.

Кроме этого, выделяются вспомогательные и смешанные алгоритмы.

Смешанный алгоритм – алгоритм, содержащий и циклы, и ветвление, и подпрограммы.

Вспомогательные алгоритмы создаются, когда возникает необходимость разбиения задачи на ряд более простых задач или когда есть необходимость многократного использования одного и того же набора действий в одном или разных алгоритмах.

Пример - песня, у которой три куплета и припев, исполняемый после каждого куплета. Алгоритм действий будет следующим:

1. Спеть 1-й куплет.

2. Спеть припев.

3. Спеть 2-й куплет.

4. Спеть припев.

5. Спеть 3-й куплет.

6. Спеть припев.

Действия, объединенные в пункт «спеть припев», трижды повторяются. Таким образом, этот алгоритм содержит набор повторяющихся одинаковых действий и возникает необходимость многократного использования одного и того же набора действий (алгоритма), следовательно такой набор действий или алгоритм можно выделить в качестве самостоятельного фрагмента. Он становится вспомогательным алгоритмом.

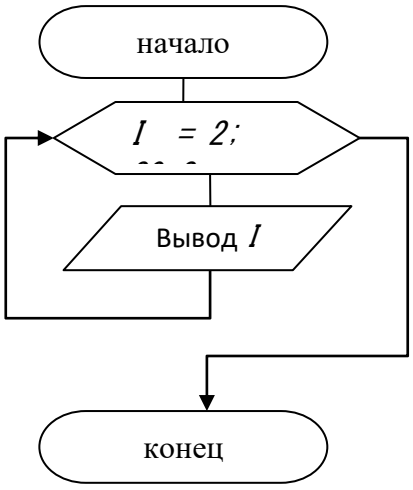
Вспомогательный алгоритм – алгоритм, по которому решается некоторая подзадача из основной задачи и который, как правило, выполняется многократно. Алгоритм может содержать несколько вспомогательных алгоритмов.

Лабораторная работа 3. Циклические алгоритмы

Упражнение 1. Циклические алгоритмы с заранее известным числом повторений (цикл с параметром)

Задача 1. Разработать алгоритм: вывести на экран четные числа от 2 до 20. Решить алгоритм с использованием программы Конструктор алгоритмов.

Решение:



Задача 2. Изменить блок-схему задачи 1, которая выводит на экран четные числа от 2 до N.

Задача 3. Изменить блок-схему задачи 1, которая выводит на экран четные числа от N до 2.

Задача 4. Составить алгоритм вычисления значения функции y=2x+5 для x=0, 1, 2, … ,10. Решить алгоритм с использованием программы Конструктор алгоритмов.

Упражнение 2. Циклические алгоритмы с заранее неизвестным числом повторений

Задача 1. Цикл с предусловием

Составить алгоритм по задаче 1 упражнения 1 с помощью цикла с предусловием и решить алгоритм с использованием программы Конструктор алгоритмов.

Задача 2. Цикл с предусловием

Составить алгоритм выбора чисел, меньше заданного числа P, в последовательности квадратов натуральных чисел (1, 4, 9, 25 и т. д.). Решить алгоритм с использованием программы Конструктор алгоритмов.

Задача 3. Цикл с постусловием

Составить алгоритм по задаче 1 упражнения 1 с помощью цикла с постусловием и решить алгоритм с использованием программы Конструктор алгоритмов.

Задача 4. Цикл с постусловием

Составить алгоритм вывода стоимости товаров в чеке до первой суммы, превышающей 1000 руб. Решить алгоритм с использованием программы Конструктор алгоритмов.

7. ПРОГРАММНЫЕ ПРОДУКТЫ ДЛЯ ПОСТРОЕНИЯ АЛГОРИТМОВ

В электронном варианте создавать алгоритмы можно даже в графическом редакторе Paint. Но в данной программе рисовать блоки придётся самостоятельно, что занимает немало времени. В настоящее время существуют специ-

альные программы, адаптированные именно на построение блок-схем. В них сделать проект можно буквально за несколько минут, и с данной задачей справится даже неопытный пользователь.

Программные продукты для построения алгоритмов в графическом виде (блок-схем) разделяются на *два вида*.

1. Построение блок-схем.

2. Построение блок-схем, их редактирование и отладка.

Кроме того, все программы разделяются на *два режима работы*:

- офлайн (скачивание и/или установка),

- онлайн (сервисы Интернет).

1 группа.

Microsoft Visio - векторный графический редактор, редактор диаграмм и блок-схем (офлайн). Программа дает возможность быстро и эффективно создавать при помощи встроенных шаблонов, трафаретов и стандартных модулей как простейшие слайды и схемы, так и очень сложные чертежи и диаграммы. Шаблон «Простая блок-схема» в Microsoft Visio содержит фигуры, которые можно использовать для наглядного представления разнообразных программ.

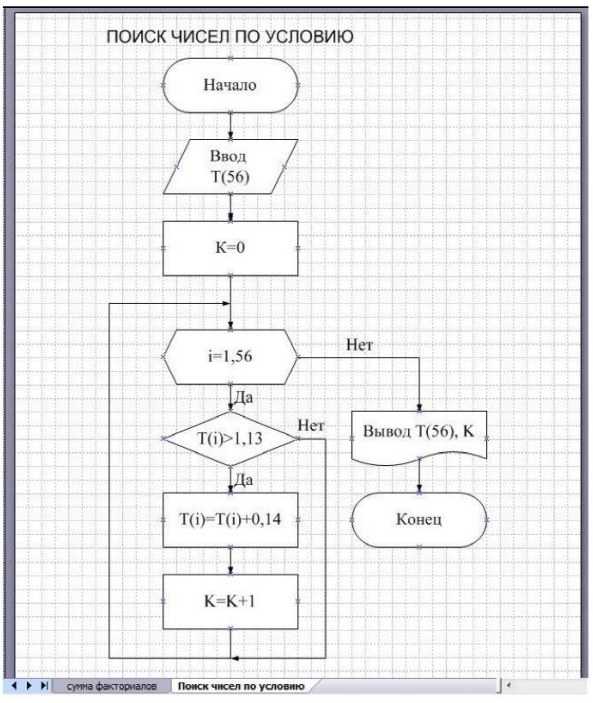


Рис. 1. Блок-схема в MS Visio

Edraw - лучшая альтернатива MS Visio для создания блок-схемы (офлайн). Программа включает все необходимые графические символы блок-схем во встроенных библиотеках. Символы строго соответствуют отраслевому стандарту. Однако их можно настроить, если необходимо, чтобы они были особенными. Имеется возможность добавлять цвета, менять стили линий и применять на рисунках быстрые стили, разработанные профессионалами.

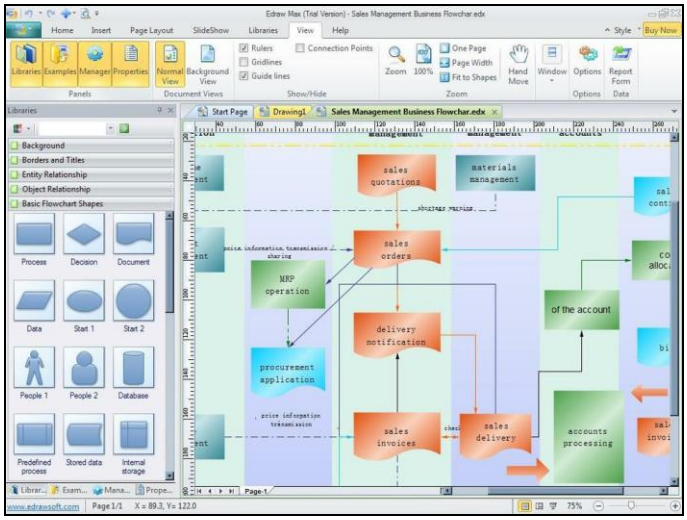


Рис. 2. Интерфейс Edraw

Считается, что данная программа для создания и редактирования блок-схем является главным конкурентом Microsoft Visio. Функционал у неё ещё более широк, но приложение позиционируется только для профессионального использования. Интерфейс максимально схож с продуктами Microsoft Office. Поддерживается свыше 10 форматов блок-схем (можно импортировать из других приложений и редактировать).

Дополнительные возможности:

- ручное рисование;
- отрисовка трёхмерной графики («объёмные» блок-схемы);
- свыше 100 шаблонов уже готовых схем.

Dia - бесплатное и полнофункциональное приложение для построения блок-схем (офлайн). Оно открыто под лицензией GPLv2.

Основные характеристики и особенности:

- простой и интуитивно понятный интерфейс;
- десятки стандартных форм, включая UML, схемы и базы данных;
- возможность добавления собственных фигур с помощью XML и SVG;
- окрашивание форм и текста в стандартные или пользовательские цвета.

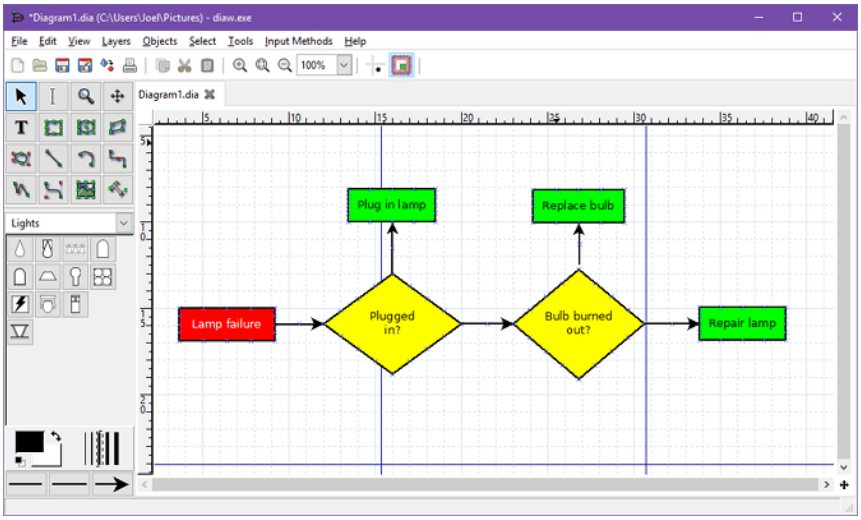


Рис. 3. Блок-схема в Dia

Algorithm Flowcharts Editor - бесплатная образовательная программа, позволяющая строить, изменять и экспортировать любые блок-схемы.

Данная программа разработана российским преподавателем. Интерфейс, соответственно, доступен только русскоязычный. Функционал простой и больше адаптирован для создания блок-схем в учебных целях: для школьников и студентов.

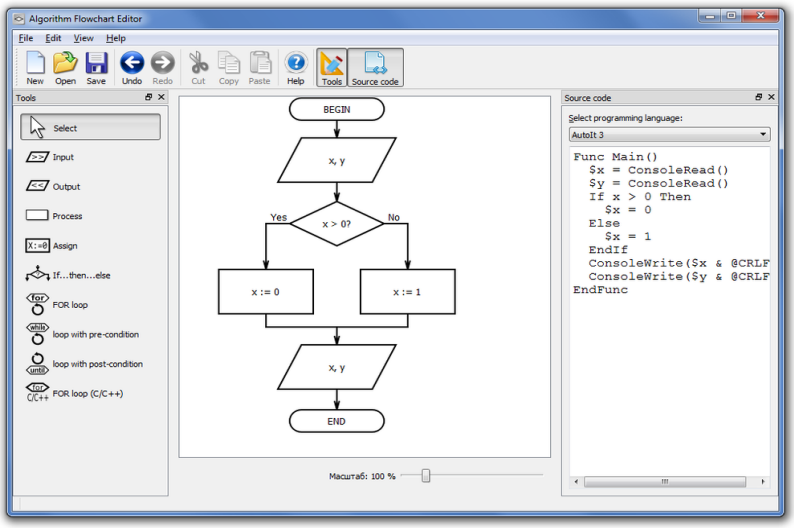


Рис. 4. Интерфейс Algorithm Flowcharts Editor

Но на текущий момент разработка и поддержка приложения прекращены. Тем не менее, Algorithm Flowcharts Editor нормально работает и в Windows 10 (и даже запускается через Wine в Linux-дистрибутивах).

Lucidchart - платформа для создания блок-схем (онлайн).

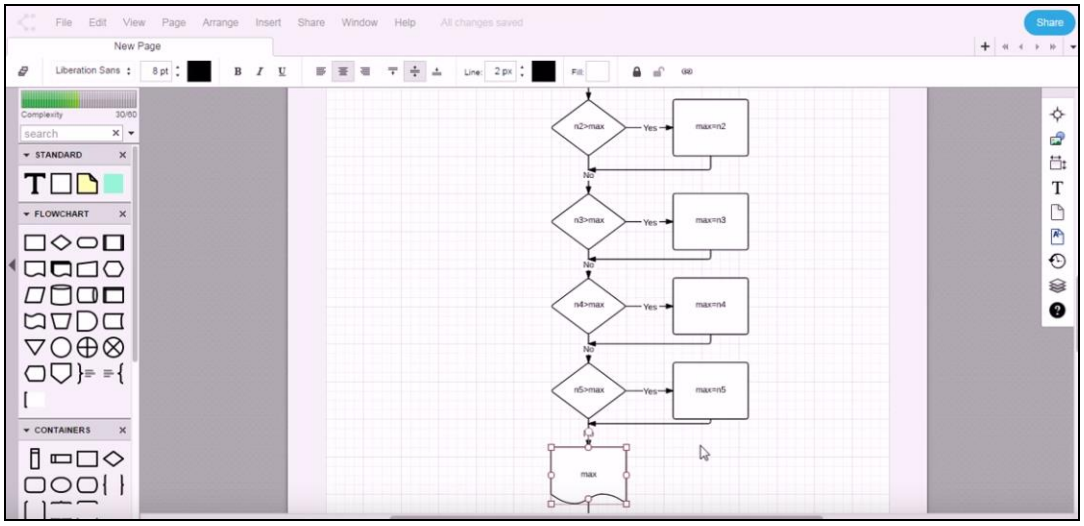


Рис. 5. Интерфейс Lucidchart

В бесплатной программе имеются элементы схематизации любых типов процессов, которые подойдут как новичкам, так и профессионалам в области блок-схем.

Draw.io – это онлайн-сервис по созданию блок-схем (онлайн). Он бесплатный, позволяет рисовать и сложные схемы, и графики, и диаграммы (как «с нуля», так и с использованием шаблонов).

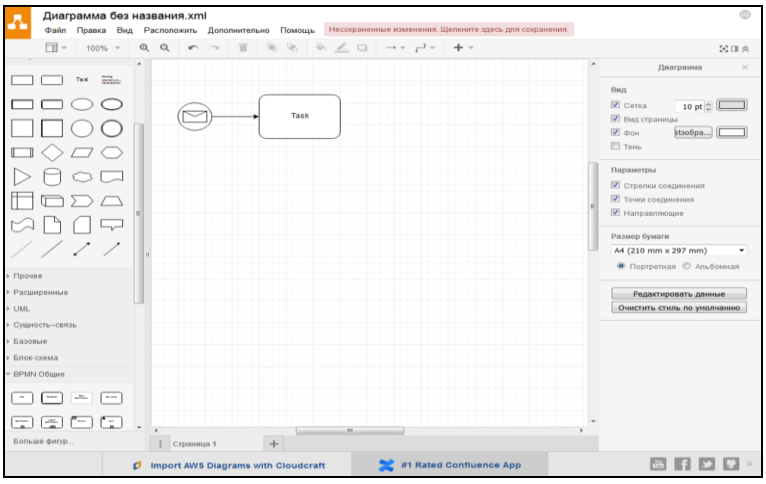


Рис. 6. Интерфейс Draw.io

Главные преимущества Draw.io:

- над схемой может работать сразу несколько пользователей (есть функция предоставления доступа к редактированию);
- автоматическая выгрузка проектов в Google Drive (если для входа использован Google аккаунт).

Из недостатков следует отметить: сервис работает онлайн, то есть требуется доступ к интернету, и некоторые функции доступны только для премиум-аккаунтов, которые активируются платно.

yEd Graph Editor - это современный инструмент для построения блок-схем, диаграмм, деревьев, сетевых графиков и многого другого. Имеется возможность загрузить приложение в виде JAR-файла или EXE-файла.

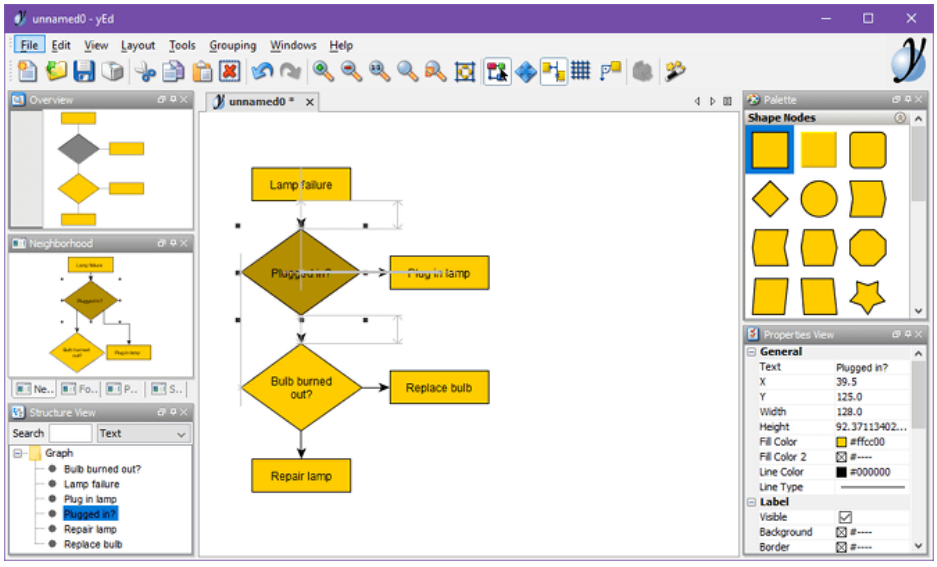


Рис. 7. Интерфейс yEd Graph Editor

ThinkComposer - это инструмент для профессионалов.

Основные характеристики и особенности:

- многоуровневые диаграммы для полного визуального выражения идей;
- композиции из множества различных графиков и диаграмм;
- генерация отчетов в формате PDF, XPS или HTML на основе данных;
- открытый исходный код и возможность расширения с помощью плагинов.

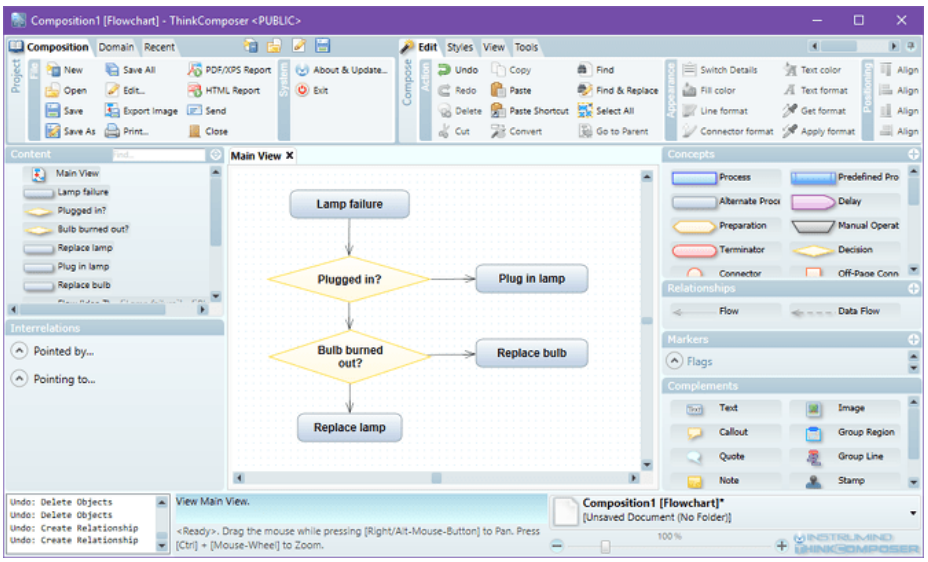


Рис. 8. Интерфейс ThinkComposer

Pencil Project - программа для тех, кому требуется быстрое и простое построение диаграмм с минимальными затратами на обучение.

Основные характеристики и особенности:

- множество встроенных фигур для всех типов графиков и интерфейсов;
- создание собственных форм или установка созданных коллекций;
- экспорт, включая PNG, SVG, PDF и HTML;
- импорт рисунков из OpenClipart.org для использования в графиках и диаграммах.

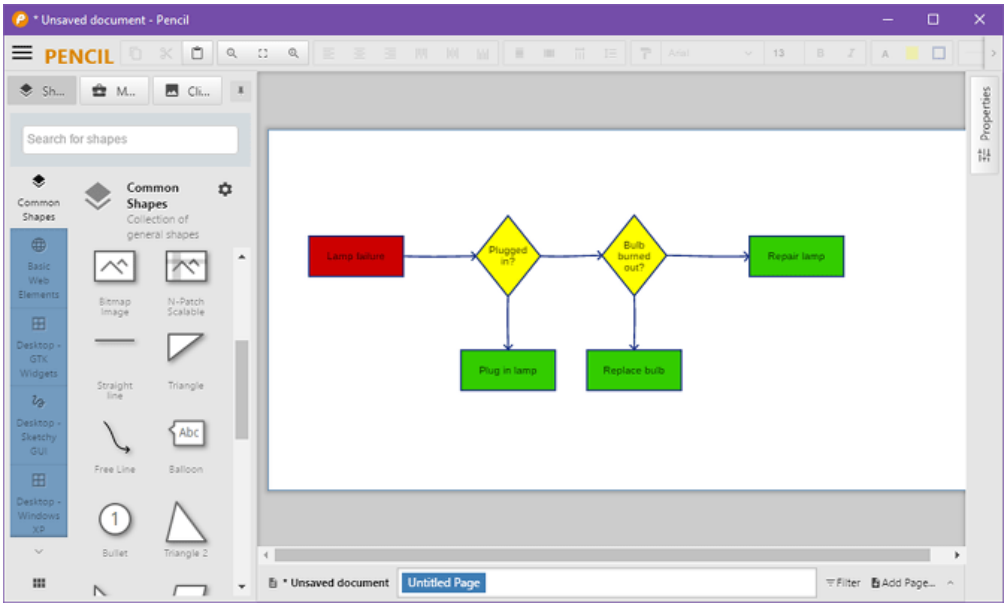


Рис. 9. Интерфейс Pencil Project

LibreOffice Draw - одна из лучших бесплатных альтернатив Microsoft Visio для обработки визуальных диаграмм (офлайн). Имеется возможность легкого добавления фигур, символов, линий соединения, текста, изображений и многого другого.

Основные характеристики и особенности:

- пользовательские размеры страниц;
- страничная карта с работой на нескольких графиках;
- современные манипуляции с объектами, включая 3D-контроллер;
- можно открыть формат Microsoft Visio.

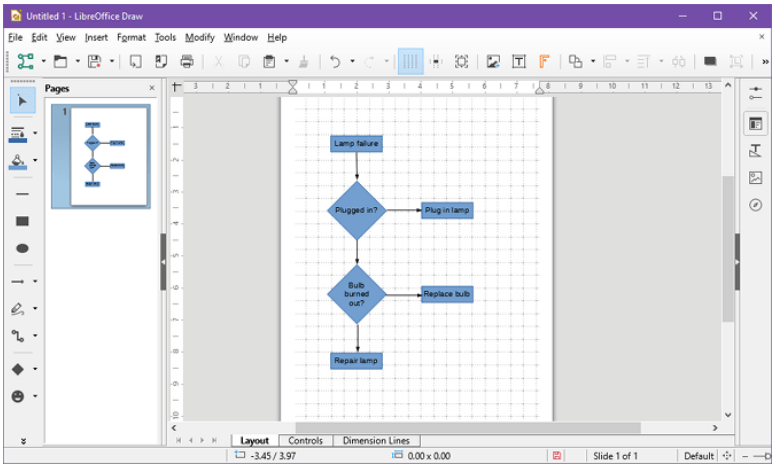


Рис. 10. Интерфейс LibreOffice Draw

Diagram Designer - простой векторный графический редактор блок-схем (офлайн). Имеет удобный интерфейс и достаточно обширный набор возможностей.

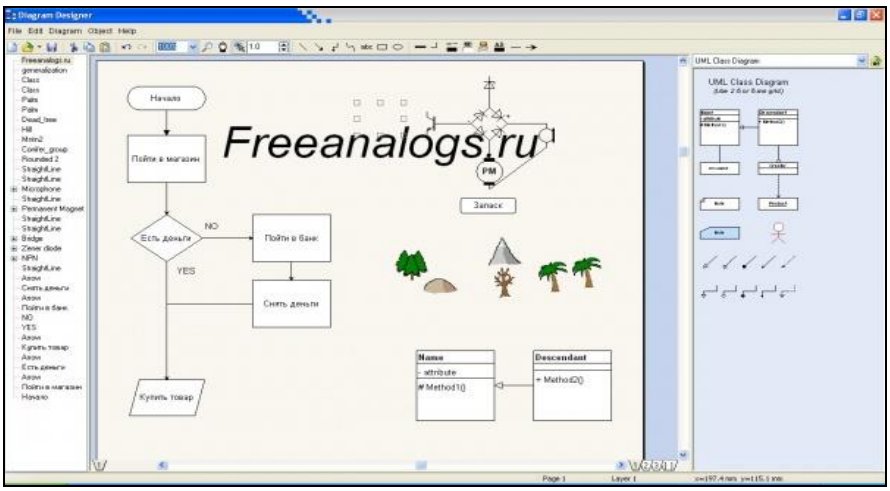


Рис. 11. Интерфейс Diagram Designer

Программа и ее исходный код распространяются бесплатно.

2 группа

Конструктор алгоритмов - инструмент предназначен для обучения структурному проектированию алгоритмов и формирования алгоритмического мышления при изучении информатики и основ программирования (офлайн). Программа позволяет создавать блок-схемы, а затем выполнять соответствующий алгоритм. Данная инструментальная среда является интерпретатором алгоритмов, представленных в виде блок-схем.

*Конструктор дает возможность разрабатывать и исполнять:

- ✓ линейные алгоритмы,
- ✓ алгоритмы с ветвлением,
- ✓ циклические алгоритмы,
- ✓ алгоритмы с процедурами,
- ✓ алгоритмы работы с двумерными и одномерными массивами.

Рис. 12. Режимы работы программы «Конструктор алгоритмов»

Порядок работы.

Работа с программой начинается с команд *Блок-схема → Новая блок-схема → Разработка*.

Слева на панели инструментов выбирается нужный блок и щелчком левой кнопки мыши вставляется на рабочее поле в соответствующей последовательности, далее каждый блок подлежит редактированию и занесению необходимых данных (ввод переменный, определение типа переменных, вычислительные формулы, задание параметра цикла, задание условия разветвляющегося алгоритма, вывод результата).

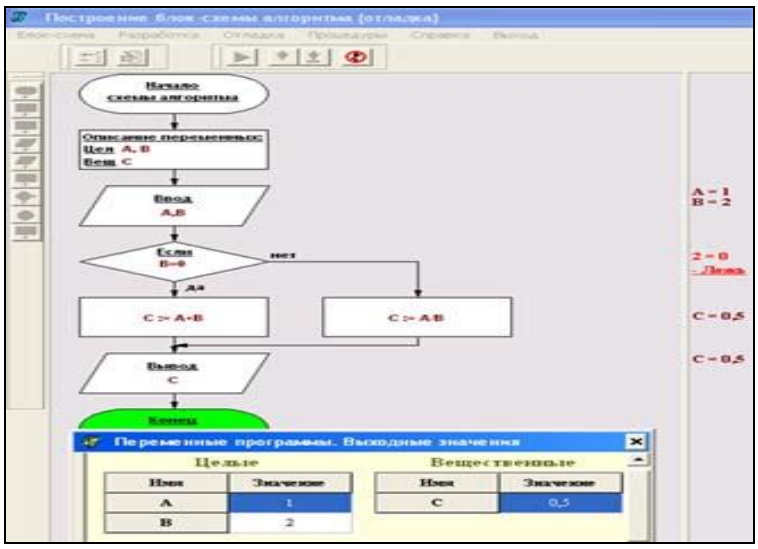


Рис. 13. Исполнение алгоритма в Конструкторе алгоритмов

В программе «Конструктор алгоритмов» есть все необходимые блоки с уже заготовленными служебными словами.

Недостатки программы:

- не работает цикл с постусловием;
- нельзя менять расположение блочных символов;
- нет функций целочисленного деления и деления с остатком.

Приведенный перечень – это далеко не все современные программные продукты для создания алгоритмов программ в графическом виде. Пользователю самое главное – это выбрать, какое приложение ему подойдёт лучше для составления блок-схем. Например, для студентов в 90% случаев хватает программы «Конструктор алгоритмов» или Algorithm Flowcharts Editor, для офисных работников – LibreOffice Draw, для инженеров – Edraw MAX.

Лабораторная работа 4. Типовые приемы алгоритмизации

Упражнение 1. Простые математические алгоритмы

Задача 1. Даны три действительных числа a, b, c. Найти максимальное из трех чисел. Решить алгоритм с использованием программы Конструктор алгоритмов.

Решение: Алгоритм решения задачи:

- Если a > b и a > c, максимальным числом будет a.
- Если c > a > b, максимальным числом будет c. Тот же результат получим, если c > b > a.
- Если b > a и b > c, максимальным числом будет b

Тестовое задание: Найти максимальное из трех чисел: 14, –67, 45. Ответ. 45

Задача 2. Табулирование функции. Составить блок-схему, которая вычисляет функцию Y=77*X+25 на интервале [0;1] с шагом 0,1. Решить алгоритм с использованием программы Конструктор алгоритмов.

Упражнение 2. Вычисление суммы или произведения

Рекурсивным называется алгоритм, который в процессе выполнения на каком-либо шаге прямо или косвенно обращается сам к себе. Как правило, в основе такого алгоритма лежит рекурсивное определение какого-то понятия.

Пример рекурсивного определения - определение факториала числа n:

n!=1*2*3*4*…*n

Сумма	Количество	Произведение
S=0	K=0	P=1
S=S+A	K=K+1	P=P*A

Задача 1. Составить блок-схему, которая вычисляет сумму 10 первых натуральных чисел. Решить алгоритм с использованием программы Конструктор алгоритмов.

Задача 2. Составить блок-схему, которая вычисляет факториал числа n:

n!=1*2*3*4*…*n

Решить алгоритм с использованием программы Конструктор алгоритмов.

Лабораторная работа 5. Алгоритмы с одномерными массивами

Массив – упорядоченная последовательность величин одного типа, обозначенная одним именем.

Образующие массив переменные называются *элементами массива***.** Упорядочение элементов массива производится их нумерацией. Каждый элемент массива обозначается *именем массива с индексом* (или несколькими индексами через запятую), заключенным в круглые скобки. Индекс определяет положение элемента массива данных относительно его начала.

Каждый массив характеризуется именем, типом и размерностью.

Имена массивам присваиваются по тем же правилам, что и переменным.

Тип массива определяется типом входящих в него величин.

Размерность массива определяется количеством изменений и максимальным числом элементов по каждому измерению.

Одномерные массивы – массивы, у которых элементы нумеруются по порядку, то есть применяется последовательная линейная нумерация.

Пример одномерного массива A(9):

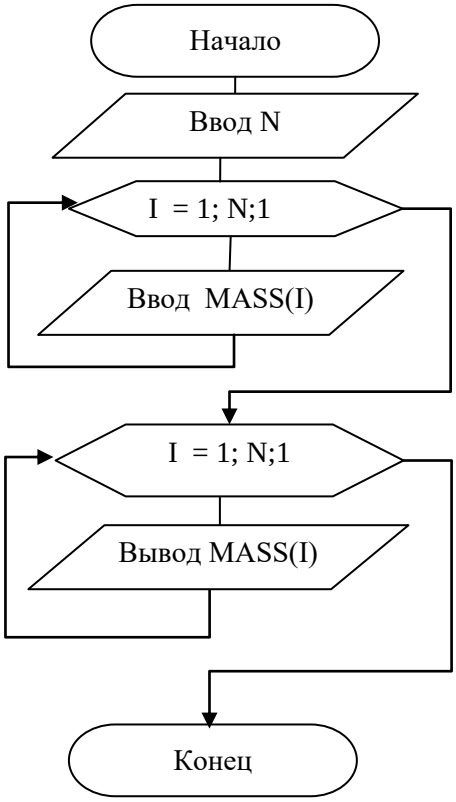
A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)	A(9)
5	7	0	25	4	13	45	259	6

Имя –A; тип – целый; размерность - 9; элемент A(7)=45

Упражнение 1. Заполнение одномерного массива

Задача 1. Составить блок-схему заполнения массив MASS(N) целыми числами с клавиатуры и вывода значений массива на экран.

Решение:



Задача 2. Составить блок-схему заполнения массива значениями функции Y=sin (X), где X меняется от 1 до 10 с шагом 1. Вывести значения массива на экран.

Упражнение 2. Обработка элементов одномерного массива

Задача 1. Составить блок-схему, которая вычисляет сумму элементов массива.

Задача 2. Задана последовательность {Xj } = x1, x2, … , xN, где j изменяется от 1 до N. Все элементы имеют разные значения. Один из элементов имеет значение P (эталон). Выполнить поиск элемента, равного P, и определить его номер..

Решение. Алгоритм состоит в следующем:

1) сравнить очередной элемент с эталоном P;

2) перейти к следующему элементу;

3) если не все элементы просмотрены, повторить, начиная с пункта 1.

Задача 3. Задана последовательность {Xj } = x1, x2, … , xN, где j изменяется от 1 до N. Длина последовательности N является фиксированной величиной. Все элементы имеют разные значения.

Найти максимальный элемент в последовательности и определить его номер.

Решение. Алгоритм состоит в следующем:

1) присвоить переменной (максимальный элемент) значение первого элемента;

2) сравнить максимальный элемент с очередным элементом последовательности - если последний больше максимального элемента, надо изменить максимальный элемент;

3) перейти к следующему элементу;

4) если не все элементы просмотрены, повторить действия, начиная с пункта 2.

Лабораторная работа 6. Алгоритмы с двумерными массивами

Двумерный массив – массив, который организован в виде таблицы или матрицы. Каждый элемент описывается двумя индексами: первый обозначает номер строки, а второй - номер столбца.

Например, массив A(3,4) зарезервирует 12 ячеек:

	Столбец 1	столбец 2	столбец 3	столбец 4
строка 1	A(1,1)	A(1,2)	A(1,3)	A(1,4)
строка 2	A(2,1)	A(2,2)	A(2,3)	A(2,4)
строка 3	A(3,1)	A(3,2)	A(3,3)	A(3,4)

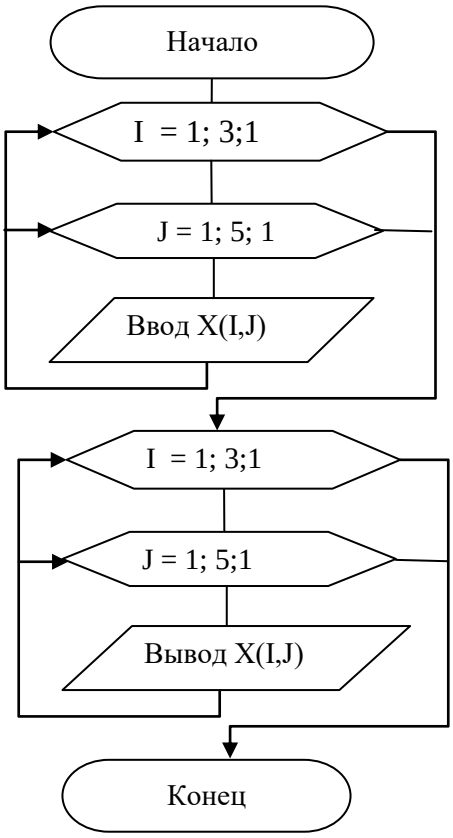
Пример: $A(3,4) = \begin{pmatrix} 3 & 5 & 7 & 9 \\ 2 & 1 & 12 & 4 \\ 6 & 2 & 5 & 8 \end{pmatrix}$

Имя – A; тип – целый; размерность 3x4; элемент A(2,3)=12.

Упражнение 1. Заполнение двумерного массива

Задача 1. Составить блок-схему заполнения массива X(3,5) целыми числами с клавиатуры и вывода значений массива на экран в виде таблицы.

Решение:



Задача 2. Составить блок-схему заполнения массива X(N,M) целыми числами с клавиатуры и вывода значений массива на экран в виде таблицы.

Задача 3. Составить блок-схему заполнения массива X(N,M) целыми числами с клавиатуры и расчета суммы элементов массива.

ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ (САМОСТОЯТЕЛЬНАЯ РАБОТА)

1. Варианты индивидуальных заданий по теме «Линейные алгоритмы».

Задание: в программе Конструктор алгоритмов составить блок-схему линейного алгоритма.

Вариант	Задание
1	В магазине продается костюмная ткань. Ее цена B руб. за квадратный метр. Рассчитать стоимость куска этой ткани длиной X метров и шириной 1м
2	Выкурив 1 сигарету, человек принимает 2 мг никотина. Вычислить сколько яда никотина примет человек за один день, выкурив X сигарет
3	По норме СанПИН площадь для одного компьютера в кабинете 3 м^2.

	Какой площадью должен быть кабинет информатики для A количества компьютеров
4	Вычислить количество оборотов колеса водяной мельницы N, необходимых для получения V м3 воды, если одним оборотом колеса мельница дает x литров воды (м3 = 1000 л)
5	Рассчитать годовую зарплату сотрудника рекламного агентства (в рублях) по формуле C=12000+250*n, где n – количество заключенных договоров в месяц
6	Вычислить плотность вещества по известным массе и объему
7	Вычислить в рублях стоимость L сантиметров ткани, если 1 метр ткани стоит S рублей
8	Вычислить скорость движения тела, если известна кинетическая энергия и масса тела (в кг)
9	Вычислить за сколько минут ракета пролетит S километров, если ракета за 1 секунду пролетает L метров
10	Вычислить скорость свободно падающего тела и пройденный им путь
11	Для перевода значения температуры по шкале Цельсия (C) в шкалу по Фаренгейту (F) используют формулу F=1,8C+32. Составить алгоритм, который переводит температуру по шкале Фаренгейта в Цельсии
12	Определить среднее значение оценок, полученных студентом в пяти опросах
13	Решить линейное уравнение в общем виде: ax + b = 0
14	Вычислить площадь треугольника, когда заданы стороны треугольника
15	Вычислить площадь треугольника, когда заданы высота и основание треугольника
16	Вычислить длину окружности

17	Вычислить площадь круга
18	Вычислить периметр прямоугольника
19	Вычислить площадь прямоугольника, ширина которого в 2 раза меньше длины
20	Вычислить периметр квадрата
21	Вычислить периметр треугольника

2. Варианты индивидуальных заданий по теме «Ветвящиеся алгоритмы».

Задание: в программе Конструктор алгоритмов составить блок-схему ветвящегося алгоритма

Вариант	**Задание**
1	Вычислить функцию $y = \frac{1}{x}$ или выдать сообщение, что функция не определена, в зависимости от введенного числа x
2	Вычислить корень квадратный наименьшего из двух, введенных с клавиатуры чисел
3	Определить четность или нечетность введенного целого числа c
4	Решить квадратное уравнение $ax^2+bx+c=0$, где $x = -\frac{b}{2a}$, если $D=b^2-4ac$ =0 и уравнение не имеет корней, если D<0
5	Ввести число. Если оно неотрицательно, вычесть 30, иначе прибавить 70
6	Вычислить сумму двух чисел, если они не равны, и произведение, если равны
7	Вычислить функцию $y = \sqrt{x^2 - 1}$ или выдать сообщение, что функция не определена, в зависимости от введенного числа x
8	Решить квадратное уравнение $ax^2+bx+c=0$, где $x = \frac{-b \pm \sqrt{D}}{2a}$, если $D=b^2-4ac >0$ и $x = -\frac{b}{2a}$, если D=0
9	Найти наибольшее из чисел (x+y)/2 и (x-y/2)для любых чисел x и y

10	Вычислить третью степень наибольшего из двух, введенных с клавиатуры чисел
11	Решить квадратное уравнение $ax^2+bx+c=0$, где $x = \frac{-b \pm \sqrt{D}}{2a}$, если $D=b^2-4ac>0$ и уравнение не имеет корней, если $D<0$
12	Найти наименьшее из чисел $(c+p)^3$ и $(c-p/3)$ для любых чисел c и p
13	Вычислить функцию $y = \ln(6m-9)$ или выдать сообщение, что функция не определена, в зависимости от введенного числа m
14	Даны три a, b, c числа целого типа. Найти количество положительных чисел.
15	Даны два числа k, e целого типа. Если числа не равны, то заменить их на 7, в противном случае заменить нулем
16	Заданы целые значения x и y. Определить z = max (x^2, y^2)
17	Вычислить стоимость покупки с учетом скидки: при покупке товара на сумму больше 500 руб. предоставляется скидка 10 %
18	Выдать информацию о знаке введенного числа (отрицательное, положительное или равно нулю)
19	Упорядочить значения двух переменных X и Y по возрастанию
20	Вычислить величину $y=\sin^2(x)+\cos(x)$ или выдать сообщение, что функция не определена, в зависимости от введенного числа x

3. Варианты индивидуальных заданий по теме «Циклические алгоритмы».

Задание: составить блок-схему циклического алгоритма.

Вариант	**Задание**
1	Составить алгоритм вывода таблицы соответствия между весом в фунтах и весом в килограммах для значений от 1 до N фунтов (1 фунт = 0,453 кг).
2	Вычислить среднее значение целых чисел от 10 до N, если N вводится с клавиатуры

3	Вычислить сумму $\frac{1}{1^5}+\frac{1}{2^5}+...+\frac{1}{N^5}$
4	Вычислить значение функции y=2x^2-3x+5 на отрезке (-3, 3) с шагом 0,2
5	Ввести число с клавиатуры. Вычитать от него 1, пока оно не станет равным 0
6	Вычислить значение функции Y= 5*x^2 -4x+11 на отрезке (-5,5) с шагом 1,5
7	Вычислить сумму квадратов первых N натуральных чисел
8	Вычислить количество нечетных чисел от 15 до N
9	Вычислить среднее геометрическое N произвольных чисел. Среднее геометрическое есть корень степени N из произведения N чисел.
10	Вычислить средний балл при поступлении в институт по результатам N экзаменов, которые вводятся с клавиатуры
11	Вычислить произведение F=cos5*cos10*cos15*…*cos N, если N вводится с клавиатуры
12	Вычислить средний рост N человек, данные о которых вводятся с клавиатуры
13	Вычислить сумму $R=\frac{1}{\sqrt{1}}+\frac{2}{\sqrt{2}}+\frac{3}{\sqrt{3}}+\cdots+\frac{N}{\sqrt{N}}$, если N вводится с клавиатуры
14	Вычислить сумму нечетных чисел от 20 до N, если N вводится с клавиатуры
15	Составить алгоритм вывода таблицы стоимости поездки на такси в рублях, в зависимости от расстояния (1 км стоит 1,5$, доллар равен курсу по ЦБ)
16	Вывести 7 первых чисел, так чтобы каждое последующее было в 2 раза больше предыдущего.

17	Одноклеточная амеба каждые 3 часа делится на 2 клетки. Определить, сколько амеб будет через 3, 6, 9, 12,…,. N часов, если N вводится с клавиатуры
18	Ввести четное число с клавиатуры. Разделить его на 2 до тех пор, пока оно не станет равно 1.
19	Вычислить сумму S=2+4+ ...+2*N, если N вводится с клавиатуры
20	Ввести число с клавиатуры. Прибавлять к нему 5, до тех пор пока оно не станет больше самого себя в 10 раз.
21	Ввести число с клавиатуры. Прибавлять к нему 5, до тех пор пока оно не станет больше самого себя в 10 раз.

4. Варианты индивидуальных заданий по теме «Одномерные массивы».

Задание: составить блок-схему с одномерным массивом.

Вариант	**Задание**
1	Подсчитать количество элементов не больших заданного числа A
2	Найти минимальный элемент массива
3	Подсчитать количество отрицательных элементов в массиве
4	Подсчитать сумму положительных элементов в массиве
5	Подсчитать произведение первых пяти элементов в массиве
6	Подсчитать сумму квадратов элементов в массиве
7	Подсчитать среднее арифметическое содержащихся в массиве чисел
8	Подсчитать сумму элементов массива, лежащих в пределах от 2 до 22
9	Заменить все четные элементы массива нулями
10	Вывести 9 элементов массива в обратном порядке по 3 числа в строке
11	Получить массив, в котором каждый элемент создается из исходного массива делением соответствующего элемента на его индекс

12	Представить в виде массива последовательность первых 20 чисел Фибоначчи, если x_1=1, x_2=2, а каждый последующий элемент равен сумме двух предыдущих
13	Даны два массива одинаковой размерности. Получить новый массив, сложив массивы поэлементно
14	Даны два массива с количеством элементов 5 и 10 соответственно. Получить массив из 15 элементов, составленный добавлением первого в конец второго
15	В массиве поменять местами значения 1-го и 2-го элемента, 3-го и 4-го и т.д.
16	Выдать в строку значения каждого третьего элемента массива.
17	Заменить все отрицательные элементы массива их квадратами
18	Подсчитать количество элементов не меньших заданного числа B
19	Найти максимальный элемент массива
20	Подсчитать количество положительных элементов в массиве
21	Подсчитать сумму отрицательных элементов в массиве
22	Подсчитать произведение последних пяти элементов в массиве
23	Подсчитать количество элементов в массиве равных нулю
24	В массиве 10 букв – С, Ф, О, И, К, Л, О, И, Л, Н. Вывести на экран слово, образованное буквами с четными индексами, и слово, образованное буквами с нечетными индексами.

5. Варианты индивидуальных заданий по теме «Двумерные массивы».

Задание: составить блок-схему с двумерным массивом.

Вариант	**Задание**
1	Вычислить сумму и число элементов каждого столбца матрицы.
2	Вычислить сумму и число элементов матрицы, находящихся под главной диагональю и на ней.

3	Заменить отрицательные элементы матрицы нулями. Вывести матрицу на печать.
4	Найти в каждой строке максимальный элемент и поместить их на место первого элемента соответствующей строки. Вывести матрицу на печать.
5	Транспонировать матрицу и вывести на печать элементы главной диагонали и диагонали, расположенной под главной диагональю.
6	Для целочисленной матрицы найти для каждой строки число элементов, кратных 5.
7	В массиве поменять местами первую и последнюю строки. Вывести матрицу на печать.
8	Вывести на экран номера строк массива, сумма элементов которых четна.
9	Заменить отрицательные элементы матрицы нулями, а положительные элементы матрицы единицами. Вывести матрицу на печать.
10	Найти число элементов массива, меньших значения C, а для элементов, больших C, найти их среднее арифметическое.
11	Поменять местами наибольший и наименьший элементы массива. Вывести матрицу на печать.
12	Заменить отрицательные элементы матрицы их квадратами, подсчитать количество таких элементов. Вывести матрицу и количество на печать.
13	Вычислить сумму и число элементов каждой строки матрицы.
14	Вычислить сумму и число элементов матрицы, находящихся над главной диагональю.
15	Заменить положительные элементы матрицы единицами. Вывести матрицу на печать.

16	Найти в каждой строке минимальный элемент и поместить его на место первого элемента соответствующей строки. Вывести матрицу на печать
17	Транспонировать матрицу и вывести на печать элементы главной диагонали и диагонали, расположенной над главной.
18	Для целочисленной матрицы найти для каждого столбца число элементов, кратных 2, и определить наименьший элемент из полученных результатов.
19	В массиве поменять местами первый и последний столбец. Вывести матрицу на печать.
20	Вывести на экран номера столбцов массива, сумма элементов которых нечетна.
21	Найти число элементов массива, больших значения К, а для элементов, больших К, найти их сумму. Вывести результаты на печать
22	Дан массив с символьными элементами. Отсортировать элементы так, чтобы слова в первой строке были расположены по алфавиту
23	Дан двумерный массив, в котором каждая строка состоит из четырех символов. Отсортировать элементы в строках так, чтобы слова были расположены по алфавиту
24	Дан массив с символьными элементами. Отсортировать элементы так, чтобы слова в первом столбце были расположены по алфавиту

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислить основные этапы компьютерного решения задач.
2. Раскрыть понятие «Алгоритмизация»
3. Что такое «алгоритм»?
4. Перечислить основные свойства алгоритмов.
5. Какие способы описания алгоритмов существуют?
6. В чем особенность графического способа описания алгоритмов?

7. Что такое «данные»?

8. Перечислить типы данных.

9. Линейные структуры алгоритмов.

10. Разветвляющиеся структуры алгоритмов

11. Циклические структуры алгоритмов. Цикл с неизвестным заранее числом повторений

12. Циклические структуры алгоритмов. Цикл с заданным числом повторений

13. Вспомогательные и смешанные алгоритмы

14.Перечислить обозначения для логических связок (операций)

15.Таблицы истинности логических выражений.

16. Какие программы для исполнения алгоритмов используются?

17. Каково назначение программы Конструктор алгоритмов?

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. 15 онлайн-сервисов для создания блок-схем [Электронный ресурс]: - Режим доступа: https://kj.media/kissel/15-onlajn-servisov-dlya-sozdaniya-blok-shem/

2. Волобуева Т.В. Информатика. Основы алгоритмизации: учеб. пособие. Воронеж: Воронежский государственный архитектурно-строительный университет, ЭБС АСВ, 2019. - 183 с. - ISBN 978-5-7731-0740-8. - Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. - URL: http://www.iprbookshop.ru/93316.html

3. Основы алгоритмизации [Электронный ресурс]: - Режим доступа: https://sites.google.com/site/udinany/vycislitelnye-masiny-sistemy-i-seti/lekcii/modul-1/lekcia-1-osnovy-algoritmizacii

4. Программа «Конструктор алгоритмов» [Электронный ресурс]: - Режим доступа: http://school-collection.edu.ru/catalog/res/fff3a9b4-5a73-445a-a617-624b63d4b8a6/?fullView=1&from=a30a9550-6a62-11da-8cd6-0800200c9a66&

5. Разумавская Е.А. Алгоритмизация и программирование [Электронный ресурс]: практ. пособие. Электрон. текстовые данные. - СПб.: Санкт-Петербургский юридический институт (филиал) Академии Генеральной прокуратуры РФ, 2015. - 49 с. - Режим доступа: http://www.iprbookshop.ru/65427.html

6. Тюльпинова Н.В. Алгоритмизация и программирование: учебное пособие. Саратов: Вузовское образование, 2019. - 200 с. - ISBN 978-5-4487-0470-3. - Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. - URL: http://www.iprbookshop.ru/80539.html

СОДЕРЖАНИЕ

Учебное издание

Ульянова Наталья Дмитриевна

Основные принципы алгоритмизации

Учебно-методическое пособие
по дисциплине «Алгоритмизация и программирование»

Компьютерный набор Ульянова Н.Д.

Редактор Осипова Е.Н.

Подписано к печати 17.11.2020 г. Формат 60x84 1/16.
Бумага печатная. Усл. п. л. 3,25. Тираж 100 экз. Изд. № 6735.

Издательство Брянского государственного аграрного университета
243365 Брянская обл., Выгоничский район, с. Кокино, Брянский ГАУ