**Практическое задание 1.**

**Основы алгоритмизации**

Цель работы ‒ изучение правил исследования прикладных задач и описание алгоритмов на языке блок-схем.

**ПЛАН**

[1. Краткое теоретическое введение 1](#_Toc18262520)

[2. Описание алгоритмов 2](#_Toc18262521)

[3. Базовые алгоритмические конструкции 3](#_Toc18262522)

[4. Линейные алгоритмы 3](#_Toc18262523)

[5. Разветвляющиеся алгоритмы 5](#_Toc18262524)

# 1. Краткое теоретическое введение

**Алгоритм**, это точное описание последовательности действий, которое ведет от произвольного набора исходных данных (возможных для данного алгоритма) к достижению результата, полностью определяемого этими данными. Алгоритм предназначен к выполнению некоторым исполнителем.

Алгоритм должен обладать определенными свойствами, наличие которых гарантирует получение решения задачи исполнителем.

**Дискретность**. Решение задачи должно быть разбито на элементарные действия. Запись отдельных действий реализуется в виде упорядоченной последовательности отдельных *команд*, образующих *дискретную* структуру алгоритма. Это свойство отражено в определении алгоритма.

**Понятность**. На практике любой алгоритм предназначен для определенного исполнителя, следовательно, любую команду алгоритма исполнитель должен уметь выполнить.

**Определенность** (**детерминированность**). Каждая команда алгоритма должна определять однозначные действия исполнителя. Результат их исполнения не должен зависеть от факторов, не учтенных в алгоритме явно. При одних и тех же исходных данных алгоритм должен давать стабильный результат.

**Массовость**. Разработанный алгоритм должен давать возможность получения результата при различных исходных данных для однотипных задач.

Например, пользуясь алгоритмом решения квадратного уравнения, можно находить его корни при любых значениях коэффициентов.

Свойство массовости полезное, но не обязательное свойство алгоритма, так как интерес представляют и алгоритмы, пригодные для решения единственной задачи.

**Результативность** (**конечность**). Это свойство предполагает обязательное получение результата решения задачи за конечное число шагов. Под решением задачи понимается и сообщение о том, что при заданных значениях исходных данных задача решения не имеет.

Смысл условий дискретности, понятности и определенности ясен: их нарушение ведет к невозможности выполнения алгоритма. Остальные условия не столь очевидны. Для сложных алгоритмов выполнить исчерпывающую проверку результативности и корректности невозможно. Это равносильно полному решению задачи, для которой создан алгоритм, вручную.

Можно сформулировать общие правила, руководствуясь которыми следует записывать алгоритм решения задачи.

1. Выделить величины, являющиеся исходными данными для задачи.

2. Разбить решение задачи на такие команды, каждую из которых исполнитель может выполнить однозначно.

3. Указать порядок выполнения команд.

4. Задать условие окончания процесса решения задачи.

5. Определить, что является результатом решения задачи в каждом из возможных случаев.

Хотя алгоритмы обычно предназначены для автоматического выполнения, они создаются и разрабатываются людьми. Поэтому первоначальная запись алгоритма обычно производится в форме, доступной для восприятия человеком.

# 2. Описание алгоритмов

Самой простой является *словесная* форма записи алгоритмов на естественном языке. В этом виде алгоритм представляет собой описание последовательности этапов обработки данных, изложенное в произвольной форме. Словесная форма удобна для человеческого восприятия, но страдает многословностью и неоднозначностью.

Когда запись алгоритма формализована частично, то используется *псевдокод*. Он содержит как элементы естественного языка, так и формальные конструкции, описывающие базовые алгоритмические структуры. Эти конструкции называются *служебными словами* Формального определения псевдокода или строгих правил записи алгоритмов в таком формате не существует.

*Графическая* форма представления алгоритма является наиболее компактной. Алгоритм изображается в виде последовательности связанных между собой блоков, каждый из которых соответствует выполнению одного или нескольких действий. Графическое представление алгоритма называется *блок-схемой*. Блок-схема определяет структуру алгоритма.

Графические обозначения блоков стандартизованы. Некоторые из часто используемых блоков представлены в таблице ниже по тексту.

Отдельные блоки соединяются линиями потоков информации. Направление линий сверху вниз или слева направо принимается за основное.

Алгоритм, записанный на любом языке программирования, называется программой. При использовании этих языков запись алгоритма абсолютно формальна и пригодна для выполнения компьютером. Отдельная конструкция языка программирования называется оператором. Программа – это упорядоченная последовательность операторов.

Таблица 1 – Изображение основных блоков на блок-схеме

|  |  |
| --- | --- |
| Обозначение блока | Пояснение |
|  | Процесс (вычислительное действие, реализованное операцией присваивания). |
|  | Выбор (проверка условия, реализующая условный переход). |
|  | Начало, конец алгоритма. |
|  | Ввод исходных данных, вывод результатов. |
|  | Модификация (для организации циклов с параметром). |

# 3. Базовые алгоритмические конструкции

Структурной элементарной единицей алгоритма является простая команда, обозначающая один элементарный шаг переработки или отображения информации. Простая команда на языке схем изображается в виде функционального блока «процесс», который имеет один вход и один выход. Из команд проверки условий и простых команд образуются составные команды, имеющие более сложную структуру, но тоже один вход и один выход.

Алгоритм любой сложности может быть представлен комбинацией трех базовых структур:

* следование;
* ветвление (в полной и сокращенной форме);
* цикл (с предусловием или постусловием).

Характерной особенностью этих структур является наличие у них одного входа и одного выхода.

# 4. Линейные алгоритмы

Базовая структура «следование» означает, что несколько операторов выполняются последовательно друг за другом, и только один раз за время выполнения программы. Структура «следование» используется для реализации задач, имеющих *линейный* алгоритм решения. Это означает, что такой алгоритм не содержит проверок условий и повторений, действия в нем выполняются последовательно, одно за другим.

**Пример 1.** Построить блок-схему алгоритма вычисления высот треугольника со сторонами *a*, *b*, *c* по формулам:

;

;

,

где  – полупериметр.

Для того чтобы не вычислять три раза одно и тоже значение, введем вспомогательную величину: .

|  |  |
| --- | --- |
|  | 1. Ввод значений сторон треугольника.  2. Вычисление полупериметра.  3. Вычисление вспомогательной величины *t*.  4. Вычисление высот, опущенных на стороны *а, b, c*.  5. Вывод результатов. |

**Пример 2**

Глубину колодца, скважины, шахты можно измерить, зная физические законы. Достаточно бросить камень и засечь время падения *t*. Требуется при заданном времени падения определить расстояние от сруба колодца до воды.

Входные данные: время полета камня, обозначено *t*. Неизвестно заранее, нужно ввести.

Формула решения:

Выходное данное – глубина. В формуле обозначено переменной *h*.

Константа: константа всемирного тяготения всегда неизменна, поэтому ее значение следует задать или в числовом выражении в теле формулы, или ввести именованное обозначение.

Описание алгоритма: задать константу *g*, ввести исходное данное, применить формулу, вывести решение.

**Упражнение 1. Линейный алгоритм**

Сформулирована прикладная задача. Выделить исходные данные, определить и обозначить результат, составить блок-схему алгоритма.

1. Прямоугольной формы строительный блок имеет ширину *w*, высоту *w* и глубину *l*. Плотность материала изготовления равна *D*. Найти массу блока. Найти, сколько будет весить палета, на которой уложено указанное количество блоков.

2. Забор должен иметь длину *L* метров. Определить, сколько будет стоить забор из рабицы стоимостью *N1* руб. погонный метр, и сколько будет стоить забор из 3D сетки стоимостью *N2* руб. метр.

# 5. Разветвляющиеся алгоритмы

Второй базовой структурой является «ветвление». Эта структура обеспечивает, в зависимости от результата проверки условия, выбор одного из двух путей работы алгоритма, причем каждый из них ведет к общему выходу (структура ЕСЛИ–ТО–ИНАЧЕ). В частном случае может оказаться, что для одного из выбранных путей действий предпринимать не надо. Это структура ЕСЛИ–ТО.

Структура с полным ветвлением записывается так:

*Если* < условие >

*то* < группа операторов 1 >

*иначе* < группа операторов 2 >

*конец если*

Команда выполняется так: если <условие> является истинным, то выполняется <группа операторов 1>, записанная после ключевого слова *то*, если <условие> является ложным, то выполняется <группа операторов 2>, записанная после слова *иначе*.

Структура с неполным ветвлением не содержит части, начинающейся со слова *иначе*:

*Если* < условие >

*то* < серия 1 >

*конец если*

Команда выполняется так: если <условие> является истинным, то выполняется <серия 1> команд, записанная после ключевого слова *то*.

Блок-схема алгоритма с ветвлением выглядит так:

|  |  |
| --- | --- |
| Полное ветвление  Структура *Если – То – Иначе* | Неполное ветвление  Структура *Если – То* |
|  |  |

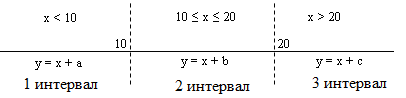
**Пример 3.** Вычислить значение функции



Дано: *x, a, b, c* – произвольные числа.

Найти: *y*.

Представим задачу графически



Так как значение переменной *x* вводится произвольно, то оно может оказаться в любом из трех интервалов.

|  |  |
| --- | --- |
|  | 1 Ввод исходных данных.  2. Проверка введенного значения. Если *х<10* (выход «Да»), то точка находится в первом интервале.  В противном случае *х ≥ 10* (выход «Нет») и точка может оказаться во втором или третьем интервале.  4. Проверка ограничения значения *х* справа (*х<20*).  Если условие выполняется (выход «Да»), то *х* находится во втором интервале, иначе *х ≥ 20* и точка находится в третьем интервале. |

**Пример 4**

Есть *V* литров жидкости. Требуется переместить ее из одной емкости в другую. Для этого есть цилиндрическое ведро радиусом *r* и высотой *h*.

Определить, можно ли это сделать за одно действие. Если нет, то определить, сколько раз нужно воспользоваться ведром.

Входные данные: объем жидкости, обозначено *V*. Неизвестно заранее, нужно ввести. Размер ведерка тоже ввести.

Выходное данное – количество ведер, которое придется вычерпать. Введем обозначение *К.*

Формула решения: результат деления нацело объема жидкости на объем ведра. Остаток следует учесть как отдельное неполное ведро.

Описание алгоритма: ввести исходные данные, найти объем ведра (ввести обозначение), применить формулу. Проверить полученное значение. Если оно равно 1, вывести ответ «одно ведро», иначе вывести реальное количество ведер.

**Упражнение 2. Алгоритмы с ветвлением**

Сформулирована прикладная задача. Выделить исходные данные, определить и обозначить результат, составить блок-схему алгоритма.

1. Рекомендуемый вес человека определяется как (*Рост*(см)–100)±5%. Ввести рост и вес, сравнить с рекомендуемым и вывести рекомендательное сообщение, например, «Необходимо похудеть на *Х* кг.».

2. Один мастер делает работу за 4 дня, и стоимость работы *К1* рублей. Другой мастер делает такую же работу за 3 дня, и стоимость работы *К2* рублей, больше, чем *К1*. Если есть *N* работ, то какого мастера выгодно нанять?