**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

**Государственное бюджетное образовательное учреждение**

**среднего профессионального образования**

**«ОРЕХОВО-ЗУЕВСКИЙ ПРОМЫШЛЕННО-ЭКОНОМИЧЕСКИЙ КОЛЛЕДЖ имени САВВЫ МОРОЗОВА»**

**Московской области**

**Методические рекомендации**

**для самостоятельной работы студентов**

**Тема: «Алгоритмы. Базовые алгоритмические структуры.**

**2014г.**

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Методические рекомендации предназначены для применения на практических занятиях и для самостоятельной работы студентов 1 курса. Содержат задания и теоретическое объяснение их выполнения, а также варианты индивидуальных заданий для самостоятельной работы студентов.

Рекомендации содержат следующие материалы:

1. Понятие алгоритма
2. Способы записи алгоритмов
3. Свойства алгоритмов
4. Схема алгоритма
5. Базовые алгоритмические структуры
6. Характеристики данных
7. Команда присваивания
8. Примеры составления алгоритмов
9. Контрольные вопросы
10. Варианты индивидуальных заданий

**КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

ЭВМ решают сегодня самые разнообразные задачи — управляют технологическими процессами, доказывают теоремы, переводят тексты с одного языка на другой, рисуют картины, сочиняют музыку, играют в шахматы. Это могут быть задачи, связанные с моделированием различных явлений и процессов, анализом результатов эксперимента, кодированием и декодированием информации, разработкой алгоритмов решения математических и иных задач, выполнением вычислений, поиском нужной информации и т. п.

Для успешного применения в практической деятельности компьютерной техники, помимо общих знаний о принципах её работы, требуется уметь строить собственные алгоритмы для решения возникающих задач.

Это пособие знакомит с основными понятиями программирования и позволяет самостоятельно составлять алгоритмы решения небольших задач.

# 

# 1. ПОНЯТИЕ АЛГОРИТМА

Процесс решения задачи представляет собой совокупность определённых действий над данными. Данными называют все величины, участвующие в решении задачи. Данные, известные перед решением, являются начальными, *исходными данными*. Результат решения задачи — это конечные, *выходные данные*. А данные, получаемые в результате обработки исходных данных и являющиеся вспомогательными данными на этапе получения выходных данных, называются *промежуточными данными*.

Решить задачу — значит получить результат. Для каждой точно сформулированной задачи всегда известно, что считать результатом.

Последовательность чётких однозначных указаний (действий), применение которых к определённым исходным данным, обеспечивает получение требуемого результата, называется *алгоритмом*.

Понятие алгоритма является одним из базовых понятий в программировании. Строгого определения алгоритма не существует. Остановимся на содержательном определении.

*Алгоритмом называется конечный набор точных и понятных предписаний (правил, инструкций, команд), позволяющих механически решать конкретную задачу из определенного класса однотипных задач.*

Алгоритм создаётся в расчёте на определённого *исполнителя*. Исполнителей алгоритма называют *формальными* исполнителями. В качестве исполнителя может выступать не только человек, но и техническое устройство — автомат, робот, ЭВМ. В первую очередь к формальным исполнителям относятся автоматические устройства, в том числе и компьютер.

Для того чтобы исполнитель мог выполнить алгоритм, необходимо, чтобы он мог выполнить каждый шаг алгоритма. Совокупность команд, которые могут быть выполнены исполнителем, называют *системой команд исполнителя*. Для правильного построения алгоритма и программы необходимо знать систему команд исполнителя.

# 

# 2. СПОСОБЫ ЗАПИСИ АЛГОРИТМОВ

Алгоритмы могут описываться различными способами, отличающимися друг от друга наглядностью, компактностью, степенью формализации. Наибольшее распространение получили способы описательный, графический и в виде программы для ЭВМ.

*Описательный способ* записи алгоритма ориентирован на исполнителя-человека. Записывается на естественном, в частности математическом, языке. Этот способ характеризуется высокой степенью понятности для исполнителя.

*Графический –* компактная форма записи в виде специальных графических символов (блоков) с указанием связей между ними. Каждый блок предписывает выполнение определённых действий. Совокупность блоков образует *схему алгоритма* (блок-схему). Графический способ записи алгоритма получил наибольшее распространение. Он характеризуется большой наглядностью и ориентирован на исполнителя-человека.

*Алгоритм в виде программы –* конечный продукт разработки алгоритма в виде программы, записанной на языке программирования.

Если задача решается с помощью ЭВМ, алгоритм решения задачи должен быть записан в понятной для машины форме, т. е. в виде программы.

# 

# 3. СВОЙСТВА АЛГОРИТМОВ

Основные свойства алгоритма: дискретность, детерминированность, массовость, результативность, конечность.

*Дискретность* означает, что путь решения задачи определён в виде последовательности шагов — чётко отделённых друг от друга предписаний (правил, инструкций, команд). Только выполнив одно предписание, можно приступить к выполнению следующего.

*Детерминированность (определённость)* означает, что на каждом шаге однозначно определено преобразование данных, полученных на предшествующих шагах алгоритма.

*Массовость* означает, что алгоритм применим к некоторому классу задач с изменяющимися в определённых пределах исходными данными.

*Результативность* означает содержательную определённость результата на каждом шаге и в итоге применения всего алгоритма. Если по каким-либо причинам невозможно решить задачу, алгоритм должен сообщить, что решения задачи не существует.

*Конечность* алгоритма гарантирует получение результата за конечное число шагов.

# 

# 4. СХЕМА АЛГОРИТМА

Представление алгоритма в виде графических символов называется *схемой алгоритма*.

Схему алгоритма следует записывать в виде условных графических символов, отображающих основные операции обработки данных, и связанные друг с другом линиями потока.

Графические символы, их размеры и правила построения схем алгоритмов должны выполняться в соответствии с государственным стандартом ГОСТ 19.701-90, согласно которому каждой группе действий соответствует блок особой формы.

Некоторые часто используемые графические символы приведены в табл.1.

При разработке алгоритма каждое действие обозначают соответствующим блоком, а последовательность выполнения — линиями со стрелками на конце. Для простоты чтения схемы желательно, чтобы линия входила в блок сверху, а выходила снизу.

Для изображения линий потока существуют следующие правила.

1. Линии должны быть параллельны линиям внешней рамки схемы алгоритма (границам листа)
2. Направление линии сверху вниз или слева направо принимается за основное и стрелками не обозначается, а в остальных случаях направление линии обозначается стрелками.
3. Изменение направления линии производится под углом 900.

Если схема алгоритма состоит из нескольких частей, расположенных на одной странице или не умещается на одном листе, то линия потока одной части заканчивается символом “Соединитель”, а линия потока следующей части схемы начинается с того же символа. Внутри символов “Соединитель” ставятся одинаковые порядковые номера, соответствующие разорванной линии потока.

| **Название блока** | **Обозначение** | **Назначение блока** | **Пример использования** |
| --- | --- | --- | --- |
| Начало, конец | b  R=0,25a  Действие | Начало или конец программы | Начало |
| Вычисление | b  a  Действие | Обработка данных (выполне­ние операций, в результате которых изменяются значе­ния, форма представления или расположение данных). | А=(Х+2)/(У+1) |
| Условие | a  b  Условие | Выбор направления выполне­ния алгоритма в зависимости от истинности или ложности некоторых условий. | A>B  нет  да |
| Ввод, вывод | 0,25a  a  b  Данные | Ввод или вывод информации. | Ввод числа N |
| Подготовка | b  а  Действия | Организация счётного цикла (начало цикла). Применяется в том случае, когда известно число повторений цикла. | K=1,5 |
| Предопреде­лённый процесс | а  Имя | Вызов процедур (использова­ние ранее созданных и от­дельно описанных алгорит­мов) | Нахождение корня |
| Соединитель | D=0,5a  Имя | Маркировка разрывов линий. | 21 |
| Межстраничный соединитель | 0,5а  0,6а  0,2а |  |  |
| Комментарий | Комментарий  0,5а  а | Пояснения к действиям | N-размерность массива |
| Линии потока |  | Указание последовательности связей между символами |  |

# 5. БАЗОВЫЕ АЛГОРИТМИЧЕСКИЕ СТРУКТУРЫ

В теории программирования доказано, что для записи любого сколь угодно сложного алгоритма достаточно трёх базовых структур.

*Линейная структура* — все действия выполняются последовательно друг за другом (рис. 1). На практике редко удаётся представить схему алгоритма решения задачи в виде линейной структуры, так как задачи содержат различные условия или требуют многократного повторения вычислений.

Действие 1

Действие 2

Рис. 1. Линейная алгоритмическая структура.

*Разветвляющаяся структура* — в зависимости от выполнения некоторого логического условия вычислительный процесс должен идти по одной или другой ветви (рис. 2). Условие — это логическое выражение, по которому происходит выбор направления выполнения алгоритма. В зависимости от значения логического выражения выполнение алгоритма идёт либо по левой, либо по правой ветви. Неполное ветвление применяется в тех случаях, если при выполнении одной из ветвей никаких изменений не происходит.

|  |  |  |
| --- | --- | --- |
| а)  Условие  Действие 1  Действие 2  да  нет | Условие  Действие  да  нет  б) | Условие  Действие  да  нет  в) |

Рис. 2. Разветвляющаяся алгоритмическая структура:  
а) полное ветвление, б) и в) неполное ветвление

*Циклическая структура* содержит многократно выполняемые участки вычислительного процесса, называемые циклами (рис. 3). Использование циклов позволяет существенно сократить схему алгоритма. Различают циклы с заданным и неизвестным числом повторений, характеризующиеся последовательным приближением к исходному значению с заданной точностью.

*Цикл с постусловием* (с последующим условием) (рис. 3, а) служит для организации циклов с заранее неизвестным числом повторений. Цикл данного типа всегда выполняется, по крайней мере, один раз, так как проверка условия завершения цикла производится после выполнения тела цикла (блока «Действие»). Такой цикл называют ещё структурой с условием завершения цикла.

*Цикл с предусловием* (с предыдущим условием) (рис. 3, б) служит для организации циклов с заранее неизвестным числом повторений. Однако в отличие от цикла с последующим условием может не выполнится ни разу.

*Цикл с параметром* (рис. 3, в) используется для организации циклического повторения некоторого фрагмента программы в случаях, когда известно число повторений тела цикла.

|  |  |  |
| --- | --- | --- |
| Действие  Условие  да  нет  а) | б)  Действие  Условие  да  нет | i=iH до iК  шаг di  Действие  в) |

Рис. 3. Циклическая алгоритмическая структура:  
а) цикл с постусловием, б) цикл с предусловием, в) цикл с параметром

# 

# 6. ХАРАКТЕРИСТИКИ ДАННЫХ

Исполнителем алгоритмов работы с данными может быть человек или специальное техническое устройство, например компьютер. Такой исполнитель должен обладать памятью для хранения данных.

Данные — это отдельный информационный объект, который имеет *имя, тип и значение*.

*Имя* объекта в алгоритме неизменно, фиксировано, уникально. Имена устанавливает автор алгоритма. Имена в программировании называются *идентификаторами*. Идентификаторы состоят, как правило, из латинских букв и цифр. Общепринято, что первым символом в идентификаторе должна быть латинская буква, за которой могут следовать другие латинские буквы и цифры. Например, N5, Y2, FIO. Буквы русского алфавита в идентификаторах, как правило, не допускаются. Рекомендуется выбирать имена, отражающие физическую суть объекта, например, PLAN (план), SUMMA (сумма).

*Тип* данных определяет множество значений, которые могут принимать данные, и множество действий, которые можно выполнить с этими данными. Основные типы данных — это *числовые* и *символьные* (текстовые).

Данные бывают постоянными (константа) и переменными.

*Константа* не изменяет своего значения в ходе выполнения алгоритма. Константа может обозначаться числом, например, 10, 5.2 или символами, например, «Иванов», «π».

**🕊** *Значение константы обычно определено в условии задачи и известно до начала разработки алгоритма!*

*Переменные* данные — это основные элементы, которыми манипулируют команды программы. *Переменные* данные могут изменять своё значение в ходе выполнения алгоритма. Переменные данные всегда обозначаются символическим именем, например, Х, A1 и т. п. *Именами переменных* обозначаются ячейки памяти, в которые будут записываться данные этого типа. Эти ячейки памяти могут получать данные, предоставлять данные для обработки и получать изменённые данные в ходе исполнения программы.

**🕊** *К моменту использования переменной в алгоритме её значение должно быть определено!*

# 7. КОМАНДА ПРИСВАИВАНИЯ

*Выражение* — это запись, определяющая последовательность действий над данными. Выражение может содержать константы, переменные, знаки операций, функции. Например:



Команда *присваивания* — команда исполнителя, в результате которой переменная получает новое значение. Формат команды:

**<имя переменной>=<выражение>**

Исполнение команды присваивания происходит в следующем порядке: Сначала вычисляется «выражение», затем полученное значение присваивается переменной.

**Пример 1.** Что происходит при выполнении команд присваивания ,  и?

Рассмотрим первую команду. Допустим, что переменная *A* имеет значение 5. Это означает, что в ячейки памяти по имени *A* хранится число 5. Выполнение команды  означает, что из ячейки памяти по имени *A* будет извлечено число 5, к нему будет добавлено число 1, а результат вычисления 6 будет помещён в ячейку с именем указанным в левой части команды *B*. Таким образом, переменная *B* получит значение равное 6.

Во втором случае происходит следующее. Допустим, что переменная *B* имеет то же значение 5. Это означает, что в ячейки памяти по имени *B* хранится число 5. Выполнение команды  означает, что из ячейки памяти по имени *B* будет извлечено число 5, к нему будет добавлено число 2, а результат вычисления 7 будет помещён в ячейку с именем указанным в левой части команды *B*. Следовательно, переменная *B* получит новое значение равное 7.

В третьем случае происходит копирование значения из ячейки с именем *A* в ячейку с именем *B.* Допустим, что переменная *А* имеет значение 1, а *В* — 3. После выполнения команды  содержимое ячейки *B* перезаписывается на новое значение, взятое из *A*, и принимает новое значение равное 1. Содержимое ячейки *A* не изменяется командой и остаётся тем же, что и до исполнения команды — 1.

**Пример 2.** Чему будет равен *Z* после выполнения всех операторов присваивания?

*X=15*

*Y=2*

*Z=(X-3\*Y^2)+7*

Выполнение:

* переменной *X* присвоить значение 15;
* переменной *Y* присвоить значение 2;
* вычислить значение правой части третьего оператора присваивания   
  (X-3\*Y^2)+7. Получится 10;
* переменной *Z* присвоить значение 10.

*Ответ*. После выполнения операторов присваивания значение переменной *Z* будет равно 10.

**Пример 3.** Даны значения переменных *a = 1, b = 2, c = 3*. Какими будут значения этих переменных после выполнения следующих трёх операторов:

*a = b*

*b = c*

*c = a*

Выполнение:

* переменной *a* присвоить значение 2;
* переменной *b* присвоить значение 3;
* переменной *c* присвоить значение 2.

*Ответ*. После выполнения операторов присваивания значение переменных будет a = 2, b =3, с =2.

### **8. ПРИМЕРЫ СОСТАВЛЕНИЯ АЛГОРИТМОВ**

### **Алгоритмы линейной структуры**

**Пример 1.**

*Математическая формулировка задачи.*

Определить площадь треугольника по формуле Герона



где a, b, c — длины сторон треугольника;

— полупериметр треугольника.

*Описательный алгоритм решения задачи.*

1. Ввод с клавиатуры исходных данных *a, b, c*.
2. Вычисление *p* по формуле.
3. Вычисление s по формуле .
4. Вывод результата s на экран монитора.

*Графический алгоритм решения задачи.*

Блок-схема алгоритма представлена на рис. 7. На семе блоки расположены в той последовательности, в которой они должны выполняться Любая перестановка блоков приведёт к невозможности решения задачи.

Начало

Ввод *a, b, c*





Конец

Вывод s

*a, b, c —* длины сторон треугольника

*p —* полупериметр сторон треугольника

*s* — вычисление площади треугольника

Вычисление площади треугольника по формуле Герона

Рис. 7. Блок-схема алгоритма вычисления площади треугольника по формуле Герона.

**Пример 2.**

*Формулировка задачи.*

Опытный рабочий выполняет задание за *a* дней, а вместе с учеником — за *b* дней. Сколько времени потребуется одному ученику для выполнения всего задания?

*Описательный алгоритм решения задачи.*

1. Ввод с клавиатуры исходных данных *a, b*.
2. Вычисление производительности рабочего *pr* по формуле.
3. Вычисление производительности рабочего и ученика *pru* по формуле.
4. Вычисление производительности ученика *pu* по формуле.
5. Вычисление времени выполнения задания учеником по формуле 
6. Вывод результата *t* на экран монитора.

*Графический алгоритм решения задачи.*

Блок-схема алгоритма представлена на рис. 8.

Начало

Ввод *a, b*

Конец

Вывод *t*



*a —* дн. рабочего

b — дн. рабочего и ученика

*pr* — производительность рабочего

*pru —* производительность рабочего и ученика

*pu —* производительность ученика

*a —* дн. рабочего

*b* — дн. рабочего и ученика

*t —* дн. ученика на выполнение задания

Рис. 8. Блок-схема алгоритма вычисления срока выполнения задания учеником.

**Пример 3.**

По представленной блок-схеме (рис. 9) получить результат вычисления алгоритма для заданных значений *a=10, b=-3, c=1*. Должен получиться ответ: *y=197.*

Начало

Ввод *a, b, c*

Конец

Вывод *y*



Рис. 9. Блок-схема алгоритма для вычисления значения *y*.

### 

### **Алгоритмы разветвляющейся структуры**

Ход решения подавляющего большинства задач может быть неоднозначен. На каком-то из этапов решения возникает необходимость выбора того или иного пути решения, в зависимости от выполнения или невыполнения некоторого условия. Подобного рода алгоритмы называются разветвляющимися.

Разветвляющимися называются алгоритмы, в которых последовательность выполнения операторов определяется некоторыми условиями.

**Пример 4.**

*Математическая формулировка задачи.*

Вычислить значение функции .

Для удовлетворения свойств массовости и результативности алгоритма необходимо, чтобы при любых исходных данных (значениях *x* и *y*) был получен результат или сообщение о том, что задача не может быть решена при заданных значениях исходных данных. Действительно, если *x* или *y* равны, то задача не может быть решена, так как деление на ноль невозможно. Поэтому в алгоритме необходимо предусмотреть вывод информации для случая, когда вычисление *z* невозможно. Такой вычислительный процесс можно описать следующим выражением:

вычислить , если ;

вывести сообщение , если .

*Описательный алгоритм решения задачи.*

1. Ввод с клавиатуры исходных данных *x, y*.
2. Проверка условия . Если условие выполняется, то вывести сообщение, что , в противном случае вычислить .
3. Вывести результат вычисления *z* на экран.

*Графический алгоритм решения задачи.*

Блок-схема алгоритма представлена на рис. 10.

Проверка условия 

Начало

Ввод x*, y*



Конец

Вывод 

да

нет

Вывод *z*



Вычисление функции 

Рис. 10. Блок-схема алгоритма решения задачи.

**Пример 5.**

*Математическая формулировка задачи.*

Составить алгоритм вычисления функции *y(x)*, при произвольных значениях *x*:



 если ;

 если ;

 если .

*Описательный алгоритм решения задачи.*

1. Ввод с клавиатуры любого значения *x*.
2. Проверка условия . Если условие выполняется, то вычислить значение *y* по формуле  в противном случае достаточно проверить условие .
3. Если условие  выполняется, то вычислить значение *y* по формуле  в противном случае .
4. Вывести результат вычисления *y* на экран.

*Графический алгоритм решения задачи.*

Схема алгоритма дложна иметьтри ветви. Сначала проверяется выполнение условия , по которому определяется выбор только одного из трёх выражений для вычисления значения переменной *y*. Для выбора одного из оставшихся двух выражений достаточно проверить условие . Тогда вычисление по формуле  будет соответствовать ветке «нет», а по формуле  — ветке «да». Вычисление по каждой из трёх ветвей завершается переходом к блоку вывода *y*. Блок-схема алгоритма представлена на рис. 11.

Проверка условия 

Начало

Ввод x



Конец

Вывод *y*

да

нет

Вычисление функции *y(x)*с условием



да

нет



Проверка условия 

Рис. 11. Блок-схема алгоритма вычисления значения функции *y(x*) с условием.

**Пример 6.**

По представленной блок-схеме алгоритма (рис. 12) проверить результат вычисления *y* для заданных значений *x*. Результаты вычислений представлены в табл. 2.

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
| **x** | -20 | 0 | 10 |
| **y** | 30 | 0 | 301 |

Начало

Ввод x



Конец

Вывод *y*

да

нет



да

нет



Рис. 12. Блок-схема алгоритма вычисления *y* из примера 6.

### 

### **Алгоритмы циклической структуры**

Болшинство практических задач требует многократного повторения одних и тех же действий, т. е. повторного использования одного или нескольких операторов.

Циклом называется многократно исполняемый участок алгоритма (программы). Соответственно циклический алгоритм — это алгоритм содержащий циклы.

Различают два типа циклов: с известным числом повторений и с неизвестным числом повторений.

**Пример 7.**

*Математическая формулировка задачи.*

Вычислить значения функции  при *x* изменяющимся от 0,1 до 1 с шагом 0,1.

Для удовлетворения свойств массовости обозначим начальную точку диапазона (0,1) за *x0*, конечную точку (1) — за *xk*, а шаг изменения значения *x* — за *h*.

*Описательный алгоритм решения задачи.*

1. Ввод с клавиатуры исходных данных: начальное значение для *x* — *x0*; конечное значение — xk; шаг изменения *x* — *h*.
2. Присвоить *x* начальное значение *x=x0*.
3. Вычислить *z* по формуле .
4. Вывести на экран рзультат вычисления *z.*
5. Изменить *x* путём прибавления к нему шага изменения параметра .
6. Проверить условие окончания вычислений *z* (выхода из цикла) . Если условие выполняется, то перейти к пункту 3 данного описания для вычисления нового значения *z*; если же условие не выполняется, то заканчиваем вычисления (выход из цикла).

*Графический алгоритм решения задачи.*

Алгоритм может быть представлен в двух вариантах.

*Вариант 1.*

Блок-схема алгоритма представлена на рис. 13.

Начало

Ввод *x0, xk, h*

Конец

Вывод *z*



Вычисление функции 

Ввод начального значения x0, конечного xk и шага h

*x* присвоить  
начальное значение

Вычисление функции 



да

нет

Увеличить *значение x* на *h*

Проверка условия окончания цикла

Рис. 13. Блок-схема алгоритма вычисления значений функции *z(x)* на заданном интервале.

*Вариант 2.*

Воспользуемся тем, что нам известно число повторений цикла, которое определяется как . Следовательно можно использовать блок «начало цикла», который выполняет все функции, необходимые для организации цикла. В этом случае блок-схема алгоритма (рис. 14) становится более компактной и наглядной.

Начало

Ввод *x0, xk, h*

Конец

Вывод *z*



Вычисление значений функции 

Ввод начального значения x0, конечного xk и шага h

*x* автоматически изменяется от *x0* до *xk* с шагом *h*

*x=x0, xk, шаг h*

Рис. 14. Блок-схема алгоритма вычисления значений функции *z(x)* на заданном интервале с использованием блока «начало цикла».

**Пример 8.**

*Формулировка задачи.*

Сборка готовок полуботинок состоит из девяти операций. На первую операцию рабочий тратит 3,5 мин, на каждую последующую на 0,5 мин больше. Определить сколько времени рабочий затрачивает на весь процесс сборки, а также, сколько изделий рабочий изготовит за 7 час работы?

Для удовлетворения свойств массовости обозначим время выполнения первой операции за *t*, а шаг изменения времени выполнения операций — за *h*.

*Описательный алгоритм решения задачи.*

1. Ввод с клавиатуры исходных данных: время выполнения первой операции t, шаг изменения времени операций — h.
2. Присвоить переменной *s,* обозначающей общее время сборки изделия значение *t,* а переменной *n*, обозначающей количество операций на сборку одного изделия значение 1.
3. Проверить условие окончания сборки изделия (выхода из цикла) n<=9. Если условие выполняется, то перейти к пункту 4 данного описания; если же условие не выполняется, то переходим к пункту 5.
4. Перменным n, t и s присваиваем новые значения: *t = t + h, s = s + t, n = n + 1*.
5. Вычисляем количество собранных изделий за 7 часов работы по формуле: *y = 7 \* 60 / s.*
6. Вывести на экран рзультат вычислениq *y, n.*

*Графический алгоритм решения задачи.*

*Вариант 1*

Начало

Ввод *t, h*

Конец

Вывод  
s, y



Вычисление времени сборки и количества изделий

*t —* время одной операции;

*h* — приращение времени

*s* — время сборки одного изделия;

n — количество операций



да

нет

Вывод результатов



*у —* количество изделий

Рис. 15. Блок-схема алгоритма вычисления времени сборки изделия и количества изготовленных изделий по варианту 1.

**12. Контрольные вопросы**

1. Дать определение алгоритма.
2. Зачем нужно графическое представление алгоритма в виде схемы алгоритма?
3. В чём состоит разница между унифицированными структурами “ветвление полное” и “ветвление неполное (обход)”?
4. Перечислите особенности структур “цикл с предварительным условием”, “цикл с последующим условием” и “цикл с параметром”.
5. Чем характеризуются линейные, разветвлённые и циклические алгоритмы?
6. Если ли ошибки в написании следующих операторов присваивания? Если есть, то укажите на ошибку.
   * 3+6=Z
   * W=5+8
   * 5+8
   * S+R=76
   * v=(x+y+z)/3
   * F+F-5
   * h=h\*2
   * x=x\*s
   * D=E=1
   * A+B=C-D
7. Понятие переменной и константы.
8. Оператор присваивания.
9. Определение алгоритма. Перечислите свойства алгоритма. Виды алгоритмов. Способы записи алгоритмов.
10. Синтаксис арифметических выражений в алгоритмическом языке.
11. Что такое исполнитель алгоритма. Типы исполнителей. Возможность автоматизации деятельности человека. Примеры.
12. Что такое полный набор исходных данных для решения задачи?
13. Линейная алгоритмическая структура. Примеры.
14. Алгоритмическая структура «ветвление». Команда ветвления. Примеры.
15. Алгоритмическая структура «цикл». Команда повторения. Примеры.
16. Алгоритмическое программирование. Основные способы организации действий в алгоритмах.

**13. Варианты индивидуальных заданий**

**Вариант 1**

1. Составить программу для вычисления длины окружности.

2. Найти произведение цифр заданного четырехзначного числа.

3. Найти площадь равнобедренной трапеции с основаниями *а* и *b* и углом *α* при большем основании *а.*

4. Идет *k-я* секунда суток. Определить, сколько полных часов и полных минут прошло к этому моменту.

5. Заданы площади круга и квадрата. Определить, поместится ли квадрат в круге.

6. Найти max{min(*a*, *b), m*in(*c,* *d)}.*

**Вариант 2**

1. Составить программу для вычисления периметра прямоугольника.

2. Вычислить периметр и площадь прямоугольного треугольника по длинам двух его катетов.

3. По координатам трех вершин треугольника найти его периметр.

4. Составить программу для решения квадратного уравнения.

5. Дано трехзначное число. Проверить, будет ли сумма его цифр четным числом.

6. Определить, делителем каких чисел *а, b, с* является число *k*.

**Вариант 3**

1. Составить программу для вычисления площади квадрата.

2. Найти длину стороны равностороннего треугольника, если известна его площадь.

3. Составить программу, запрашивающую год рождения пользователя, год рождения его мамы и печатающую, сколько лет было маме в момент рождения пользователя.

4. Заданы три числа. Определить, есть ли среди них хотя бы одна пара одинаковых чисел.

5. Написать программу вычисления разности наибольшего и наименьшего из трех чисел.

6. Заданы площади круга и квадрата. Определить, поместится ли круг в квадрате.

**Вариант 4**

1. Составить программу для вычисления площади равностороннего треугольника.

2. Найти среднее арифметическое трех чисел.

3. Известно, что плата за детский сад за месяц составляет 550 рублей (за 22 дня). Рассчитать, сколько нужно заплатить за месяц, если ребенок посещал детский сад *n* дней.



4. Составить программу для вычисления значения данного выражения:

5. Определить, одного ли знака два числа, вводимые с клавиатуры.

6. Найти количество положительных чисел среди четырех целых чисел. Если количество положительных чисел больше количества отрицательных, то найти максимальное среди положительных, иначе найти минимальное среди отрицательных. Определить количество чисел, равных нулю.

**Вариант 5**

1. Составить программу для вычисления площади прямоугольного треугольника.

2. Дана сумма начисленной заработной платы. Из этой суммы необходимо удержать 12 % на подоходный налог, 1 % на профсоюзный налог, 1 % на пенсионный налог и добавить 45 %. Полученную сумму вывести на экран.

3. Запросить у пользователя курс доллара на сегодняшний день, затем имеющуюся у него сумму в рублях и рассчитать, сколько долларов он может купить.

4. Даны две прямые *у1=а1x+в1* и *у2=а2x+в2*. Напечатать координаты точки пересечения этих прямых или сообщить, что эти прямые параллельны.

5. Написать программу, которая позволит по заданным длинам сторон треугольника выяснить, является ли он прямоугольным.

6. Даны два угла треугольника (в градусах). Определить, существует ли такой треугольник. Определить его вид (остроугольный, прямоугольный, тупоугольный).

**Вариант 6**

1. Составить программу для вычисления объема прямоугольного параллелепипеда.

2. Найти стоимость покупки, состоящей из 2 кг конфет, 1 булки хлеба и 0,75 л молока. Цены вводятся с клавиатуры.

3. Найти разность между первыми двумя и последними двумя цифрами заданного четырехзначного числа.

4. Заданы радиус круга *r* и сторона квадрата *а*. Определить, какая фигура имеет большую площадь.

5. Проверить, является ли введенное с клавиатуры число кратным 2 и 5.

6. Написать программу, определяющую, равны ли площади двух треугольников, которые заданы длинами сторон.

**Вариант 7**

1. Составить программу для вычисления площади круга.

2. Длина ребра куба равна *а*. Вычислить площадь полной поверхности куба.

3. Даны два числа. Найти среднее арифметическое кубов этих чисел.

4. Найти разность между первыми двумя и последними двумя цифрами заданного шестизначного числа.

5. Определить, является ли число *а* делителем числа *b*. (Для определения остатка использовать функцию mod.)

6. Написать программу вычисления разности наибольшего и наименьшего из трех чисел.

**Вариант 8**

1. Составить программу для вычисления площади трапеции.

2. Даны два числа. Найти среднее арифметическое модулей этих чисел.

3. Найти разность между первой и последней цифрами заданного трехзначного числа.

4. Определить, имеется ли среди заданных целых чисел *а, b, с* хотя бы одно четное.

5. Написать программу, которая позволит по заданным длинам сторон треугольника выяснить, является ли он прямоугольным.

6. Даны три числа *а, b, с*. Определить, какое из них равно некоторому числу d. Если среди данных чисел таких нет, то найти max(*d — a, d — b, d — с*).

**Вариант 9**

1. Составить программу для вычисления площади параллелограмма.

2. Вычислить расстояние между двумя точками с заданными координатами.

3. Найти сумму цифр заданного двухзначного числа.

4. Проверить, превышает ли число 200 сумма квадратов двух натуральных чисел.

5. Два прямоугольника заданы длинами сторон. Определить, можно ли первый прямоугольник целиком разместить во втором.

6. Найти значение выражения.



**Вариант 10**

1. Составить программу для вычисления площади треугольника по формуле Герона.

2. Дана длина стороны равностороннего треугольника. Найти длины его высот.

3. Найти произведение цифр заданного двухзначного числа.

4. Проверить, принадлежит ли точка с заданными координатами третьей координатной четверти.

5. Написать программу, которая возводит данное число в куб, если оно является четным, увеличивает на 3 — в противном случае.

6. Даны три действительных числа *х, у, z*. Составить программу, которая позволяет определить, что больше — сумма или произведение этих чисел.