МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Гжельский государственный университет»** (ГГУ)

Колледж ГГУ

Специальность 09.02.07 Информационные системы и программирование

**Практическая работа**

по предмету «МАПО»

на тему «Обратное проектирование алгоритма»

ВЫПОЛНИЛ:

Студент группы ИСП-О-17

Белкин И.В.

ПРОВЕРИЛА:

Прокуронова А. Ю.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

п. Электроизолятор

1. год

**Введение.**

Процессор электронно-вычислительной машины, это чудо техники, умеет, тем не менее, выполнять лишь простейшие команды. Каким же образом компьютер решает сложнейшие задачи обработки информации? Для решения этих задач программист должен составить подробное описание последовательности действий, которые необходимо выполнить центральному процессору компьютера. Составление такого пошагового описания процесса решения задачи называется *алгоритмизацией* , а *алгоритмом* называется конечный набор правил, расположенных в определённом логическом порядке, позволяющий *исполнителю* решать любую конкретную задачу из некоторого класса однотипных задач. В разных ситуациях в роли исполнителя может выступать электронное или какое-либо иное устройство или человек (например, военнослужащий, охраняющий склад боеприпасов и действующий согласно алгоритмам, записанным в устав караульной службы).

**Алгоритм. Свойства алгоритма.**

Само слово «алгоритм» возникло из названия латинского перевода книги арабского математика IX века Аль-Хорезми «Algoritmi de numero Indoru», что можно перевести как «Трактат Аль-Хорезми об арифметическом искусстве индусов». Составление алгоритмов и вопросы их существования являются предметом серьёзных математических исследований.

**Свойства алгоритма.** При составлении и записи алгоритма необходимо обеспечить, чтобы он обладал рядом свойств.

*Однозначность алгоритма* , под которой понимается единственность толкования исполнителем правила построения действий и порядок их выполнения. Чтобы алгоритм обладал этим свойством, он должен быть записан командами из системы команд исполнителя.

*Конечность алгоритма* – обязательность завершения каждого из действий, составляющих алгоритм, и завершимость выполнения алгоритма в целом.

*Результативность алгоритма* , предполагающая, что выполнение алгоритма должно завершиться получением определённых результатов.

*Массовость* , т. е. возможность применения данного алгоритма для решения целого класса задач, отвечающих общей постановке задачи. Для того чтобы алгоритм обладал свойством массовости, следует составлять алгоритм, используя обозначения величин и избегая конкретных значений.

*Правильность алгоритма* , под которой понимается способность алгоритма давать правильные результаты решения поставленных задач.

*Эффективность* – для решения задачи должны использоваться ограниченные ресурсы компьютера (процессорное время, объём оперативной памяти и т. д.).

**Описание алгоритмов на естественном языке.**

Если речь идёт о составлении алгоритмов для процессора ЭВМ (электронно-вычислительной машины), исполнителем является процессор. Упрощённая модель процессора содержит устройство считывания данных, стёк (специальную оперативную память небольшого объёма, предназначенную для временного хранения данных) и арифметическое устройство, которое может выполнять арифметические действия.

Предположим, что программа, составленная для такого процессора, содержит числовые данные и символы арифметических действий над этими данными. Вот пример такой программы, предназначенной для вычисления сумм двух чисел 2 и 3:

2, 3, +

Проследим выполнение этой программы. Первая операция - считывание в стёк значения 2. Затем в стёк считывается второе значение (3). Первое значение при этом сдвигается во вторую ячейку памяти. Третий шаг выполнения программы – вычисление суммы двух считанных значений (они называются *операндами* ). Результат этой операции – значение 5 – записывается в первую ячейку стёка.

Был рассмотрен пример простейшей программы. Она является записью алгоритма решения некоторого класса задач – задач вычисления суммы двух чисел. Обозначим эти числа *a* и *b* . Тогда алгоритм можно записать следующим образом:

1. Считать число *a* .

2. Считать число *b* *.*

3. Выполнить суммирование *c* := *a* + *b* .

4. Вывести число *c* .

Это пример записи алгоритма *на естественном языке* , то есть на языке человеческого общения. Видно, что формулировка алгоритма не зависит от конкретных значений переменных *a* и *b* , поэтому его можно применять для решения достаточно большого числа сходных задач, вместе составляющих целый *класс* задач суммирования. Алгоритм описывает действия не над конкретными значениями, а над абстрактными объектами.

Основными объектами программирования являются *переменные* . Переменные в программе отличаются от переменных, используемых в записи математических формул. Несмотря на сходство терминов, правила использования переменных в программах для компьютера отличаются от правил работы с математическими переменными. Это различие необходимо уяснить. В программировании переменную можно трактовать как одну или несколько ячеек оперативной памяти компьютера, которым присвоено определённое имя. Содержимое этих ячеек может меняться, но имя переменной остаётся неизменным. В математике значение переменной в рамках определённой задачи неизменно, но меняется в других задачах из данного класса. Именно поэтому конструкция

а := а + 1

воспринимается программистом совершенно естественно, а уравнение

*a* = *a* + 1

математик сочтёт неверным. В первом случае имеется в виду вычисление суммы содержимого ячейки а и числовой константы 1 и занесение полученного результата в ту же ячейку а. Второй случай равносилен неверному тождеству 0 = 1.

Оставим алгоритм решения следующей задачи. Пусть заданы два значения *x* и *y* . Необходимо сравнить эти значения и напечатать имя большей переменной. Для этой задачи достаточно сравнить оба значения и в зависимости от результата сравнения вывести на печать символ «х» и символ «у»:

1. Ввести значение *x* .

2. Ввести значение *y* .

3. Если *x* < *y* , то напечатать «у», иначе напечатать «х».

В этом алгоритме используются *алгоритмические структуры* - *линейная последовательность операций* и *ветвление* (шаг 3, условный оператор). Последняя структура называется так потому, что после передачи в неё управления выполнение алгоритма может пойти по одному из двух возможных ветвлений. То, какая ветвь будет выбрана, зависит от выполнения условия. Линейная последовательность в данном примере состоит из блоков *ввода* /*вывода данных* .

Для записи алгоритмов использовался естественный язык. Иногда используют полуформальный язык с ограниченным словарём (часто на основе английского языка), промежуточный между естественным языком и языком программирования. Такой язык называется *псевдокодом* . Запись алгоритма на псевдокоде называется *структурным планом* . Псевдокод удобен тем, что позволяет программисту сосредоточиться на формулировке алгоритма, не задумываясь над синтаксическими особенностями конкретного языка программирования.

**Описание алгоритмов с помощью блок-схем.**

Для разработки структуры программы удобнее пользоваться записью алгоритма в виде *блок-схемы* (в англоязычной литературе используется термин *flow* *-* *chart* ). Для изображения основных алгоритмических структур и блоков на блок-схемах используют специальные графические символы. Они приведены на рисунке

https://www.bestreferat.ru/images/paper/94/69/8926994.gifhttps://www.bestreferat.ru/images/paper/95/69/8926995.gif Начало/конец алгоритма

https://www.bestreferat.ru/images/paper/96/69/8926996.gifhttps://www.bestreferat.ru/images/paper/97/69/8926997.gif Передача управления

https://www.bestreferat.ru/images/paper/98/69/8926998.gifhttps://www.bestreferat.ru/images/paper/99/69/8926999.gif Ввод данных

https://www.bestreferat.ru/images/paper/00/70/8927000.gifhttps://www.bestreferat.ru/images/paper/95/69/8926995.gif Блок вычислений

https://www.bestreferat.ru/images/paper/01/70/8927001.gifhttps://www.bestreferat.ru/images/paper/95/69/8926995.gif Начало (заголовок) цикла

https://www.bestreferat.ru/images/paper/02/70/8927002.gifhttps://www.bestreferat.ru/images/paper/03/70/8927003.gif Конец цикла

https://www.bestreferat.ru/images/paper/04/70/8927004.gifhttps://www.bestreferat.ru/images/paper/05/70/8927005.gif Ветвление

https://www.bestreferat.ru/images/paper/06/70/8927006.gifhttps://www.bestreferat.ru/images/paper/07/70/8927007.gif Вывод данных

Составим алгоритм вычисления квадратного корня из произвольного положительного вещественного числа *х*методом Герона и запишем его на естественном языке, а также в виде блок-схемы. Метод основан на многократном применении формулы:

https://www.bestreferat.ru/images/paper/08/70/8927008.gif

при

https://www.bestreferat.ru/images/paper/09/70/8927009.gif.

Числовая последовательность https://www.bestreferat.ru/images/paper/10/70/8927010.gifв пределе при https://www.bestreferat.ru/images/paper/11/70/8927011.gif сходится к искомому значению. Выполним только 5 итераций метода, считая, что при этом будет достигнута достаточно хорошая точность. Обычно десяти итераций метода Герона более чем достаточно для достижения хорошей точность расчёта. Оба варианта записи алгоритма:

|  |  |
| --- | --- |
| https://www.bestreferat.ru/images/paper/12/70/8927012.gif | 1. Ввести *х* .  2. Присвоить https://www.bestreferat.ru/images/paper/13/70/8927013.gif.  3. Присвоить https://www.bestreferat.ru/images/paper/14/70/8927014.gif.  4. Присвоить https://www.bestreferat.ru/images/paper/15/70/8927015.gif.  5. Присвоить https://www.bestreferat.ru/images/paper/16/70/8927016.gif.  6. Если https://www.bestreferat.ru/images/paper/17/70/8927017.gif, то перейти к шагу 4, иначе напечатать значение https://www.bestreferat.ru/images/paper/18/70/8927018.gif. |

А теперь займёмся самым любимым занятием школьников всех времён и народов – решением квадратного уравнения:

https://www.bestreferat.ru/images/paper/19/70/8927019.gif.

Будем полагать, что коэффициенты этого уравнения https://www.bestreferat.ru/images/paper/20/70/8927020.gif, https://www.bestreferat.ru/images/paper/21/70/8927021.gif и https://www.bestreferat.ru/images/paper/22/70/8927022.gif представляют собой вещественные числа. Простейший случай предполагает, что все коэффициенты отличны от нуля. В зависимости от знака дискриминанта квадратного уравнения

https://www.bestreferat.ru/images/paper/23/70/8927023.gif

возможны три случая:

1. Если https://www.bestreferat.ru/images/paper/24/70/8927024.gif, то имеются два различных вещественных корня, которые можно вычислить по следующим формулам:

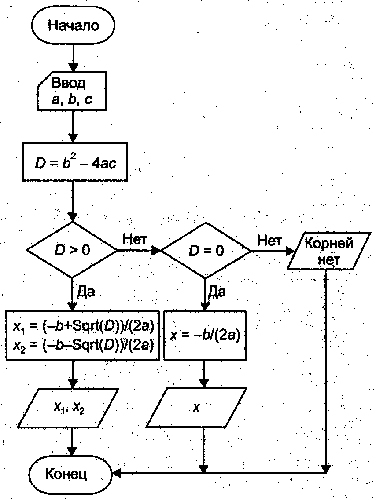
https://www.bestreferat.ru/images/paper/25/70/8927025.gif, https://www.bestreferat.ru/images/paper/26/70/8927026.gif.

2. Если https://www.bestreferat.ru/images/paper/27/70/8927027.gif, то имеется единственный корень (точнее, двукратный корень):

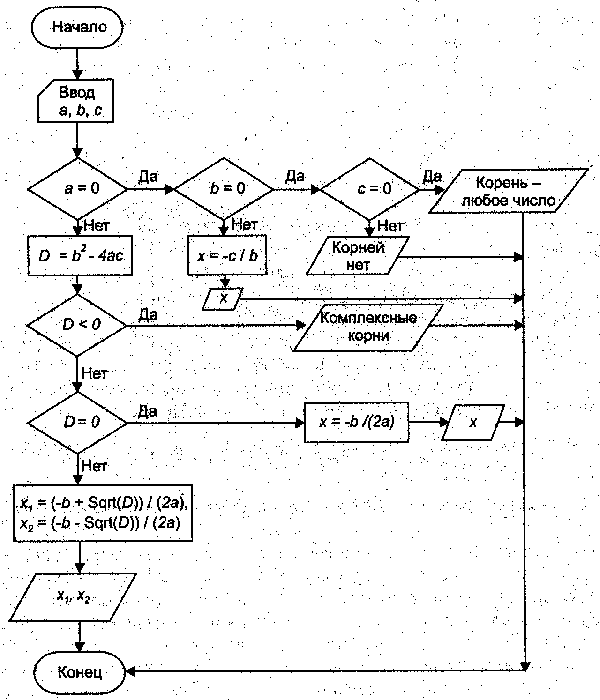
https://www.bestreferat.ru/images/paper/28/70/8927028.gif.

3. Если https://www.bestreferat.ru/images/paper/29/70/8927029.gif, то вещественных корней нет.

Блок схема алгоритма приведена на рисунке:



Следует заметить, что приведённый алгоритм предназначен для решения узкого класса задач – квадратных уравнений с «хорошими» коэффициентами. Если допустить, что коэффициенты могут принимать произвольные вещественные значения, есть опасность, что при определённых значениях коэффициента (например, https://www.bestreferat.ru/images/paper/31/70/8927031.gif) возникает аварийная ситуация (деление на ноль). Качественный алгоритм и качественная программа должны быть устойчивыми, то есть при любых входных параметрах завершение работы программы должно быть нормальным, хотя, возможно, и сопровождаться предупреждающим сообщением о некорректности входных данных. Свойством устойчивости обладает алгоритм решения квадратного уравнения, приведённый на рисунке:



Разработанный программистом алгоритм должен давать правильный ответ. Проверка алгоритма может оказаться непростым делом. В простых случаях такая проверка может быть выполнена с помощью заполнения *трассировочной таблицы* . Каждый столбец такой таблицы соответствует определённой переменной, а каждая строка – одному шагу алгоритма. Для заполнения таблицы необходимо шаг за шагом проследить выполнение алгоритма, записывая в таблицу текущие значения выбранных для трассировки переменных. Такой метод позволяет выявить логические ошибки, допущенные при составлении или записи алгоритма, и определить, верен ли окончательный ответ. Составим в качестве примера трассировочную таблицу для алгоритма Герона вычисления квадратного корня из числа 2.

|  |  |
| --- | --- |
| i | z |
| 0 | 1,00000 |
| 1 | 1,50000 |
| 2 | 1,41666 |
| 3 | 1,41421 |
| 4 | 1,41421 |
| 5 | 1,41421 |

Как видно из таблицы, уже после третьей итерации приближенное значение квадратного корня отличается от точного 1,414213 лишь в шестом знаке после запятой.

**Заключение.**

Создание алгоритма для решения задач какого-либо типа, его представление исполнителю в удобной для него форме – это творческий акт. Алгоритм может быть представлен различными способами: на разговорном естественном язык; на языке блок-схем; на языке программирования. Выбор и разработка алгоритма и численного метода решения задачи имеют важнейшее значение для успешной работы над программой. Тщательно проработанный алгоритм решения задачи – необходимое условие эффективной работы по составлению алгоритму.