1. ОСНОВЫ АЛГОРИТМИЗАЦИИ

Цель работы: изучение этапов решения задач на ЭВМ, форм записей алгоритмов, правил оформления лабораторных работ, создания, сохранения и компиляции файлов в среде *Lasarus*.

Этапы решения задачи на ЭВМ

Решение любой задачи на ЭВМ состоит из нескольких этапов, среди которых основными являются следующие:

1) постановка задачи;
2) формализация (математическая постановка задачи);
3) выбор метода решения;
4) разработка алгоритма;
5) составление программы;
6) отладка программы;
7) вычисление и обработка результатов.

Последовательное выполнение перечисленных этапов составляет полный цикл разработки.

Постановка задачи. При постановке задачи первостепенное внимание должно уделяться выяснению конечной цели и выработке общего подхода к исследуемой проблеме. Правильно сформулировать задачу иногда не менее сложно, чем ее решить.

Формализация. Формализация сводится к построению математической модели рассматриваемого явления, когда в результате анализа задачи:

1) определяются объем и специфика исходных данных;
2) вводится система условных обозначений;
3) устанавливается принадлежность решаемой задачи к одному из известных классов задач;
4) выбирается соответствующий математический аппарат.

Выбор метода решения. После того, как определена математическая формулировка задачи, надо выбрать метод ее решения. Применение любого метода приводит к построению ряда формул и к формулировке правил, определяющих связи между этими формулами. Все это разбивается на отдельные действия так, чтобы вычислительный процесс мог быть выполнен машиной.

Разработка алгоритма. Данный этап заключается в разложении вычислительного процесса на возможные составные части, установлении порядка их следования, описании содержания каждой такой части в той или иной форме.

В процессе разработки алгоритм проходит несколько этапов детализации. Первоначально составляется укрупненная схема алгоритма, в которой отражаются наиболее важные и существенные связи между исследуемыми процессами. На последующих этапах раскрываются выделенные на предыдущих этапах части вычислительного процесса.

В процессе разработка алгоритма могут использоваться различные способы его описания, отличающиеся по простоте, компактности, наглядности, степени детализации, ориентации на машинную реализацию и другим показателям. Наибольшее распространение получили:

1) словесная запись алгоритма;
2) схемы алгоритмов;
3) псевдокод (формальные алгоритмические языки);
4) структурограммы.

Составление программы. Представление алгоритма в форме, допускающий ввод в машину, относится к этапу программирования. Разработанный алгоритм задачи необходимо изложить на языке, который будет понятен ЭВМ.

Отладка программы. Первоначально составленная программа обычно содержит ошибки, и машина или не может дать ответа, или приводит неправильное решение.

Отладка начинается с того, что программа, аккуратно записанная на листе, проверяется непосредственно лицом, осуществившим программирование задачи. Выясняется правильность написания программы, выявляются смысловые и синтаксические ошибки. Затем программа вводится в ЭВМ и те ошибки, которые остались незамеченными, выявляются уже непосредственно с помощью машины.

Гарантией правильности решения может служить:

1) проверка выполнения условий задачи;
2) качественный анализ задачи;
3) перерасчет другим методом.

Для некоторых программ процесс отладки может потребовать значительно больше машинного времени, чем собственно решение.

Вычисления и обработка результатов. Только после того, как появится уверенность, что программа обеспечивает получение правильных результатов, можно приступать непосредственно к расчетам. После завершения расчетов наступает этап использования результатов в практической деятельности.

Словесная форма записи алгоритмов

Первоначально для записи алгоритмов пользовались средствами обычного языка, но с тщательно отобранным набором фраз, не допускающим синонимов, двусмысленностей и лишних слов.

Рассмотрим в качестве примера алгоритм Эвклида.

Даны два целых положительных числа, найти их наибольший общий делитель (НОД). Решение данной задачи может быть получено последовательным вычитанием из большего числа меньшего, затем из меньшего – полученного остатка, и т.д. до тех пор, пока значения чисел не сравняются.

Обозначим через M и N исходные числа, приняв в качестве их значений некоторые константы. Алгоритм может быть сформулирован следующим образом:

1) если M не равно N, то перейти к п.2, иначе перейти к п.5;
2) если M больше N, то перейти к п.3, иначе перейти к п.4;
3) от M отнять значение N и далее считать эту разность значением M; перейти к пункту 1.
4) от N отнять значение M и далее считать эту разность значением N; перейти к пункту 1.
5) считать, что НОД равен M; окончание.

Так как алгоритм предназначен для реализации на ЭВМ, то обязательно надо:

1) ввести указатели начала и конца алгоритма;
2) дополнить алгоритм указанием о вводе в ЭВМ значений M и N;
3) дополнить алгоритм указанием о выводе результата;
4) упростить запись п.3 и п.4.

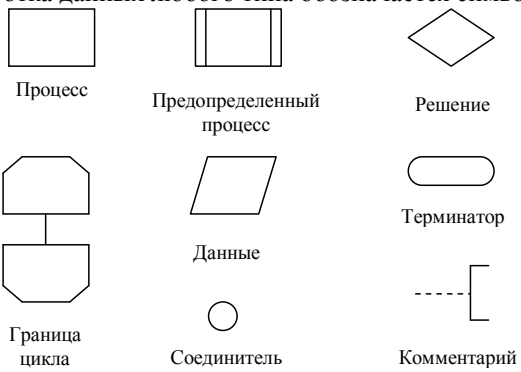
```
Начало.
1. Ввести (M, N).
2. Если M ≠ N, то перейти к п.3 иначе перейти к п.6.
3. Если M > N, то перейти к п.4 иначе перейти к п.5.
4. M = M – N; перейти к п2.
5. N = N – M; перейти к п2.
6. НОД = M.
7. Вывод (НОД).
Конец.
```

Схемы алгоритмов

Схема – это графическое представление алгоритма, дополненное элементами словесной записи. Каждый пункт алгоритма отображается на схеме некоторой геометрической фигурой – блоком, причем различным по типу выполняемых действий блокам соответствуют различные геометрические фигуры.

Символы схемы алгоритма соединяются **линиями потока**, определяющими последовательность выполнения действий. На линиях потока должны быть стрелки, показывающие направления линий. Допустимо не ставить стрелку, если линия потока идет слева направо или сверху вниз.

Обработка данных любого типа обозначается символом процесс.

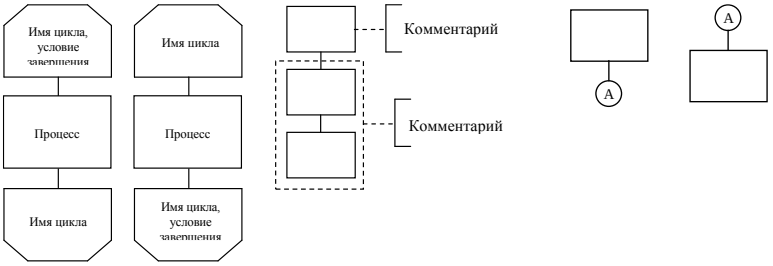


Символ **предопределенный процесс** показывает одну или несколько операций, которые определены в другом месте (в подпрограмме, модуле).

Символ **решение** отображает решение, имеющую один вход и несколько альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа.

Символ **граница цикла** состоит из двух частей и показывает начало и конец цикла.

Символы схемы алгоритма рекомендуется изображать, соблюдая одинаковые размеры для одинаковых символов. При необходимости допустимо изменять размеры символов, но нельзя параметры, влияющие на форму символов. Если требуемый текст не умещается внутри символа, то рекомендуется использовать **символ комментария.**



Символ **данные** отображает ввод-вывод данных. Символ **терминатор** показывает начало или конец схемы программы. Символ **соединитель** используется для обрыва линии потока и продолжения ее в другом месте. Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение.

Оформление лабораторных работ

Основной целью данного цикла лабораторных работ является приобретение практических навыков алгоритмизации задач и конструирования программ на языке *Pascal* в среде *Lasarus.*

Перед выполнением лабораторных работ необходимо изучить методические указания и разобрать примеры программ по каждой работе, а также ответить на контрольные вопросы. Лабораторные работы должны выполняться в следующем порядке:

1) изучить методические указания;
2) для заданного преподавателям варианта разработать схему решения задачи, разработать схему алгоритма;
3) составить программу на языке *Pascal* в соответствии с разработанной схемой алгоритма;
4) для выбранных самостоятельно данных просчитать вручную контрольный вариант, используемый в дальнейшем для отладки программы;
5) представить схему алгоритма, текст программы и результаты просчета контрольного варианта на проверку преподавателю;
6) ввести программу в ЭВМ, отладить и получить результаты просчета контрольного варианта;
7) при совпадении результатов ручного и машинного просчета контрольного варианта получить результаты решения для данных, указанных в варианте задания;
8) представить результаты решения задачи на проверку преподавателю;
9) оформить отчет о выполненной работе;

10) защитить выполненную работу перед преподавателем.

Отчет о проделанной работе должен содержать: 1) титульный лист; 2) цель работы; 3) вариант задания; 4) схему алгоритма; 5) текст программы; 6) результаты расчетов на ЭВМ, включая контрольные задания.

Образец титульного листа отчета

Министерство науки и высшего образования РФ

ФГБОУ ВО «РГРТУ» имени В.Ф. Уткина

Кафедра КТ

ОТЧЕТ

по лабораторной работе №3

«Разработка и реализация алгоритмов разветвляющейся структуры»

по курсу "Программирование и алгоритмические языки "

Выполнил: студент гр. 848
Иванов В.В.

Проверил: доц. каф. КТ.
Наумов Д.А.

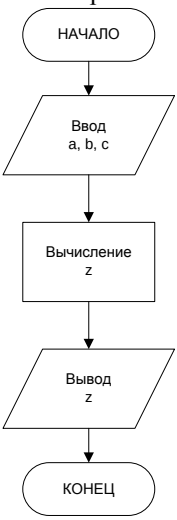
Рязань 2019

Оборот титульного листа отчета

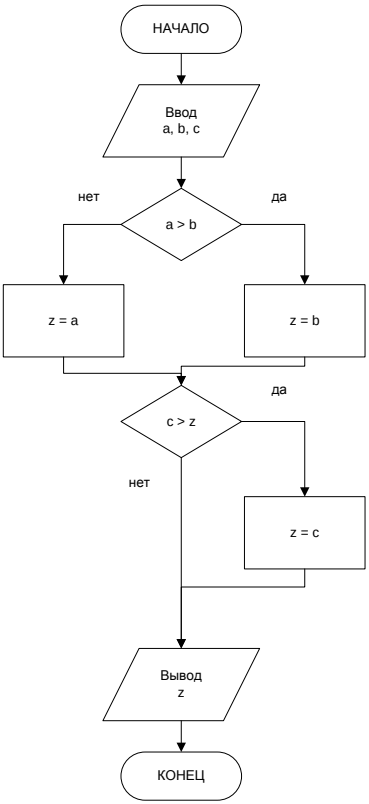
Цель работы: изучение условных операторов, оператора выбора, составного оператора и правил программирования разветвляющихся алгоритмов.

Задание. Вариант №1. Даны три различных числа a, b, c. Составить программу вычисления значения z, равного квадрату наибольшего из чисел a, b, c.

Укрупненная схема алгоритма



Детализированная схема алгоритма



Текст программы:

```
//Лабораторная работа №3, вариант 1
// Выполнил: ст. гр. 848 Иванов В.В.
// Проверил: доц. каф. КТ Наумов Д.А.
// Дата написания: 4 октября 2019
program lab03_v01;

var
    a, b, c, d: real;

begin
    writeln('Input source data');
    write('a='); readln(a);
    write('b='); readln(b);
    write('c='); readln(c);

    writeln('Source data');
    writeln('a=', a:10:4);
    writeln('b=', b:10:4);
    writeln('c=', c:10:4);

    if a > b then
       d := a
    else
       d := b;
    if c > d then
       d = c;

    d = d * d;

    writeln('Result');
    write('d = ', d:10:4 );
end.
```

Просчет контрольного варианта:

При a = 3.83, b = 1.53, c = 4.5 максимальным значением является 4.5, следовательно, z = 4.52 = 20.25 .

Расчет результатов на ЭВМ:

1) a = 3.83, b = 1.53, c = 4.5; z = 20.25
2) a = 3.83, b = 1.53, c = - 4.5; z = 14.6689
3) a = 3.83, b = 15, c = 4.5; z = 225
4) a = 11, b = 1.53, c = 4.5; z = 121

Задание к лабораторной работе

Выполнение задания состоит из следующих этапов:

1) изучить средства разработки программ на языке FreePascal (Free Pascal и Lazarus: Учебник по программированию / Е. Р. Алексеев, О. В. Чеснокова, Т. В. Кучер — М. : ALT Linux ; Издательский дом ДМК-пресс, 2010. — 440 с. : ил. — (Библиотека ALT Linux).);
2) выбрать три произвольные программы из предлагаемых в лабораторной работе (Листинг 1 – Листинг 7);
3) изучить процесс ввода и компиляции программы:
    a. в текстовом редакторе, откомпилировав при помощи компилятора fpc.exe в командной строке);
    b. в редакторе FreePascal;
    c. в среде Lazarus (создавая консольное приложение).

```
// Демонстрация операции shl
begin
  writeln('Степени двойки');
  writeln(' n         2^n');
  for var i:=0 to 30 do
    writeln(i:2,(1 shl i):12);
end.
```

Листинг 1

```
// Стандартные функции
var x: real;

begin
  write('Введите x: ');
  readln(x);
  writeln('Квадрат ',x,' равен ',sqr(x));
  writeln('Квадратный корень из ',x,' равен ',sqrt(x));
  writeln('Модуль ',x,' равен ',abs(x));
  writeln('Натуральный логарифм ',x,' равен ',ln(x));
  writeln('Синус ',x,' равен ',sin(x));
  writeln('Косинус ',x,' равен ',cos(x));
end.
```

Листинг 2

```
// Использование вспомогательных переменных
var r: real;
var r2,r4,r8: real; // вспомогательные переменные

begin
  write('Введите r: ');
  readln(r);

  r2 := r * r;
  r4 := r2 * r2;
  r8 := r4 * r4;

  writeln(r,' в степени 8 = ',r8);
end.
```

Листинг 3

```
// Вывод результатов вычислений. Используются переменные и процедура ввода
var a,b: integer;

begin
  writeln('Введите a и b:');
  readln(a,b);
  writeln;
  writeln(a,' + ',b,' = ',a+b);
  writeln(a,' - ',b,' = ',a-b);
  writeln(a,' * ',b,' = ',a*b);
  writeln(a,' / ',b,' = ',a/b);
end.
```

Листинг 4

```
// Операции div и mod
var a: integer;

begin
  write('Введите a: ');
  readln(a);
  writeln('Последняя цифра числа: ',a mod 10);
  writeln('Число без последней цифры: ',a div 10);
  writeln('Если число a четно, то 0: ',a mod 2);
end.
```

Листинг 5

```
// Операции div и mod
var a: integer := 247;

begin
  write('Цифры числа в обратном порядке: ');
  // Выводим последнюю цифру
  write(a mod 10,' ');
  // Отбрасываем последнюю цифру
  a := a div 10;
  write(a mod 10,' ');
  a := a div 10;
  write(a mod 10,' ');
end.
```

Листинг 6

```
// Генерация случайного числа
var
  r: real;

begin
  r := Random;
  writeln('Случайное вещественное в диапазоне [0,1): ',r);
end.
```

Листинг 7