

# Глава 8. Технология подготовки и решения задач с помощью компьютера

## 8.1. Какие этапы включает в себя решение задач с помощью компьютера?

Решение задач с помощью компьютера включает в себя следующие основные этапы, часть из которых осуществляется без участия компьютера.

1. **Постановка задачи:**
  - 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. **Анализ и исследование задачи, модели:**
  - анализ существующих аналогов;
  - анализ технических и программных средств;
  - разработка [математической модели](#);
  - разработка структур данных.
3. **Разработка алгоритма:**
  - выбор метода проектирования алгоритма;
  - выбор формы записи алгоритма (блок-схемы, псевдокод и др.);
  - выбор [тестов](#) и метода тестирования;
  - проектирование алгоритма.
4. **Программирование:**
  - выбор языка программирования;
  - уточнение способов организации данных;
  - запись алгоритма на выбранном языке программирования.
5. **Тестирование и отладка:**
  - синтаксическая отладка;
  - отладка семантики и логической структуры;
  - тестовые расчеты и анализ результатов тестирования;
  - совершенствование программы.
6. **Анализ результатов решения задачи** и уточнение в случае необходимости математической модели с повторным выполнением этапов 2 — 5.
7. **Сопровождение программы:**
  - доработка программы для решения конкретных задач;
  - составление документации к решенной задаче, к математической модели, к алгоритму, к программе, к набору тестов, к использованию.

## 8.2. Что называют математической моделью?

**Математическая модель** — это система математических соотношений — формул, уравнений, неравенств и т.д., отражающих существенные свойства объекта или

## явления.

**Всякое явление природы бесконечно в своей сложности.** Проиллюстрируем это с помощью примера, взятого из книги В.Н. Тростникова "Человек и информация" (Издательство "Наука", 1970).

... Обыватель формулирует математику задачу следующим образом: *"Сколько времени будет падать камень с высоты 200 метров?"* Математик начнет создавать свой вариант задачи приблизительно так: *"Будем считать, что камень падает в пустоте и что ускорение силы тяжести 9,8 метра в секунду за секунду. Тогда ..."*

— *Позвольте, — может сказать "заказчик", — меня не устраивает такое упрощение. Я хочу знать точно, сколько времени будет падать камень в реальных условиях, а не в несуществующей пустоте.*

— *Хорошо, — согласится математик. — Будем считать, что камень имеет сферическую форму и диаметр... Какого примерно он диаметра?*

— *Около пяти сантиметров. Но он вовсе не сферический, а продолговатый.*

— *Тогда будем считать, что он имеет форму эллипсоида с полуосями четыре, три и три сантиметра и что он падает так, что большая полуось все время остается вертикальной. Давление воздуха примем равным 760 мм ртутного столба, отсюда найдем плотность воздуха...*

Если тот, кто поставил задачу на "человеческом" языке не будет дальше вмешиваться в ход мысли математика, то последний через некоторое время даст численный ответ. Но "потребитель" может возражать по-прежнему: *камень на самом деле вовсе не эллипсоидальный, давление воздуха в том месте и в тот момент не было равно 760 мм ртутного столба и т.д.* Что же ответит ему математик?

Он ответит: *"Точное решение реальной задачи вообще невозможно. Мало того, что форму камня, которая влияет на сопротивление воздуха, невозможно описать никаким математическим уравнением; его вращение в полете также неподвластно математике из-за своей сложности. Далее, воздух не является однородным, так как в результате действия случайных факторов в нем возникают флуктуации колебания плотности. Если пойти ещё глубже, нужно учесть, что по закону всемирного тяготения каждое тело действует на каждое другое тело. Отсюда следует, что даже маятник настенных часов изменяет своим движением траекторию камня."*

*Короче говоря, если мы всерьез захотим точно исследовать поведение какого-либо предмета, то нам предварительно придется узнать местонахождение и скорость всех остальных предметов Вселенной. А это, разумеется, невозможно ....*

Чтобы описать явление, необходимо выявить самые существенные его свойства, закономерности, внутренние связи, роль отдельных характеристик явления. Выделив наиболее важные факторы, можно пренебречь менее существенными.

Наиболее эффективно математическую модель можно реализовать на компьютере в виде алгоритмической модели — так называемого "вычислительного эксперимента" (см. [1], параграф 26).

Конечно, результаты вычислительного эксперимента могут оказаться и не соответствующими действительности, если в модели не будут учтены какие-то важные стороны действительности.

Итак, создавая математическую модель для решения задачи, нужно:

1. выделить предположения, на которых будет основываться математическая модель;
2. определить, что считать исходными данными и результатами;
3. записать математические соотношения, связывающие результаты с исходными данными.

При построении математических моделей далеко не всегда удастся найти формулы, явно выражающие искомые величины через данные. В таких случаях используются математические методы, позволяющие дать ответы той или иной степени точности.

Существует не только математическое моделирование какого-либо явления, но и визуально-натурное моделирование, которое обеспечивается за счет отображения этих явлений средствами машинной графики, т.е. перед исследователем демонстрируется своеобразный "компьютерный мультфильм", снимаемый в реальном масштабе времени. Наглядность здесь очень высока.

### 8.3. Какие основные этапы содержит процесс разработки программ?

Процесс разработки программы можно выразить следующей формулой:

$$\boxed{\begin{array}{c} \text{РАЗРАБОТКА} \\ \text{ПРОГРАММЫ} \end{array}} = \boxed{\begin{array}{c} \text{ИЗГОТОВЛЕНИЕ} \end{array}} + \boxed{\begin{array}{c} \text{ДОКАЗАТЕЛЬСТВО} \\ \text{ПРАВИЛЬНОСТИ} \end{array}}$$

На начальном этапе работы анализируются и формулируются требования к программе, разрабатывается точное описание того, что должна делать программа и каких результатов необходимо достичь с ее помощью.

Затем программа разрабатывается с использованием той или иной технологии программирования (например, структурного программирования).

Полученный вариант программы подвергается систематическому тестированию — ведь наличие ошибок в только что разработанной программе это вполне нормальное закономерное явление. Практически невозможно составить реальную (достаточно сложную) программу без ошибок. Нельзя делать вывод, что программа правильна, лишь на том основании, что она не отвергнута машиной и выдала результаты. Все, что достигнуто в этом случае, это получение каких-то результатов, не обязательно

правильных. В программе при этом может оставаться большое количество логических ошибок. Ответственные участки программы проверяются с использованием методов доказательства правильности программ.

Для каждой программы обязательно проводятся работы по обеспечению качества и эффективности программного обеспечения, анализируются и улучшаются временные характеристики.

## 8.4. Как проконтролировать текст программы до выхода на компьютер?

Текст программы можно проконтролировать за столом с помощью просмотра, проверки и прокрутки.

- **Просмотр.** Текст программы просматривается на предмет **обнаружения описок и расхождений с алгоритмом**. Нужно просмотреть **организацию всех циклов**, чтобы убедиться в правильности операторов, задающих кратности циклов. Полезно посмотреть еще раз **условия в условных операторах, аргументы в обращениях к подпрограммам** и т.п.
- **Проверка.** При проверке программы программист по тексту программы мысленно старается восстановить тот вычислительный процесс, который определяет программа, после чего сверяет его с требуемым процессом. На время проверки нужно "**забыть**", что должна делать программа, и "**узнавать**" об этом по ходу её проверки. Только после окончания проверки программы можно "**вспомнить**" о том, что она должна делать и **сравнить** реальные действия программы с требуемыми.
- **Прокрутка.** Основой прокрутки является **имитация программистом за столом выполнения программы на машине**. Для выполнения прокрутки приходится задаваться какими-то исходными данными и производить над ними необходимые вычисления. **Прокрутка — трудоемкий процесс**, поэтому ее следует применять лишь для контроля логически сложных участков программ. **Исходные данные должны выбираться такими, чтобы в прокрутку вовлекалось большинство ветвей программы.**

## 8.5. Для чего нужны отладка и тестирование?

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

**Тестирование** (англ. test — испытание) — это испытание, проверка правильности работы программы в целом, либо её составных частей.

Отладка и тестирование — это два четко различимых и непохожих друг на друга этапа:

- при **отладке** происходит локализация и устранение синтаксических ошибок и явных ошибок кодирования;

- в процессе же **тестирования** проверяется работоспособность программы, не содержащей явных ошибок.

**Тестирование устанавливает факт наличия ошибок, а отладка выясняет ее причину.**

Английский термин *debugging* ("отладка") буквально означает "вылавливание жучков". Термин появился в 1945 г., когда один из первых компьютеров — "Марк-1" прекратил работу из-за того, что в его электрические цепи попал мотылек и заблокировал своими останками одно из тысяч реле машины.

## 8.6. В чем заключается отладка?

В современных программных системах (Turbo Basic, Turbo Pascal, Turbo C и др.) отладка осуществляется часто с использованием специальных программных средств, называемых отладчиками. Эти средства позволяют **исследовать внутреннее поведение программы**.

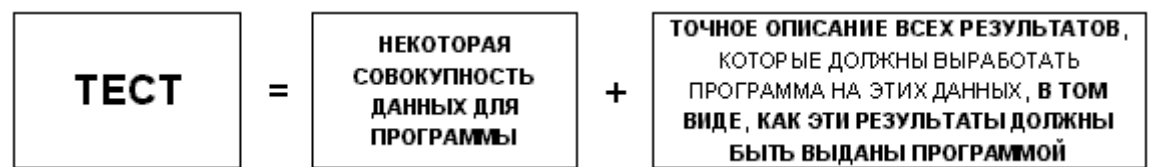
Программа-отладчик обычно обеспечивает следующие возможности:

- **пошаговое исполнение программы** с остановкой после каждой команды (оператора);
- **просмотр текущего значения любой переменной или нахождение значения любого выражения**, в том числе, с использованием стандартных функций; при необходимости можно установить новое значение переменной;
- **установку в программе "контрольных точек"**, т.е. точек, в которых программа временно прекращает свое выполнение, так что можно оценить промежуточные результаты, и др.

При отладке программ **важно помнить следующее**:

- в начале процесса отладки надо использовать **простые тестовые данные**;
- возникающие затруднения следует четко **разделять и устранять строго поочередно**;
- **не нужно считать причиной ошибок машину**, так как современные машины и трансляторы обладают чрезвычайно высокой надежностью.

## 8.7. Что такое тест и тестирование?



Как бы ни была тщательно отлажена программа, решающим этапом, устанавливающим ее пригодность для работы, является контроль программы по результатам ее выполнения на системе тестов.

**Программу условно можно считать правильной, если её запуск для выбранной системы тестовых исходных данных во всех случаях дает правильные результаты.**

Но, как справедливо указывал известный теоретик программирования Э. Дейкстра, тестирование может показать лишь наличие ошибок, но не их отсутствие. Нередки случаи, когда новые входные данные вызывают "отказ" или получение неверных результатов работы программы, которая считалась полностью отлаженной.

Для реализации метода тестов должны быть изготовлены или заранее известны эталонные результаты.

**Вычислять эталонные результаты нужно обязательно до, а не после получения машинных результатов.**

В противном случае имеется опасность невольной подгонки вычисляемых значений под желаемые, полученные ранее на машине.

## 8.8. Какими должны быть тестовые данные?

Тестовые данные должны обеспечить проверку всех возможных условий возникновения ошибок:

- должна быть испытана **каждая ветвь** алгоритма;
- очередной тестовый прогон должен контролировать нечто такое, что еще **не было проверено** на предыдущих прогонах;
- первый тест должен быть **максимально прост**, чтобы проверить, работает ли программа вообще;
- арифметические операции в тестах должны **предельно упрощаться** для уменьшения объема вычислений;
- количества элементов последовательностей, точность для итерационных вычислений, количество проходов цикла в тестовых примерах должны задаваться из соображений **сокращения объема вычислений**;
- минимизация вычислений не должна снижать надежности контроля;
- тестирование должно быть **целенаправленным и систематизированным**, так как случайный выбор исходных данных привел бы к трудностям в определении ручным способом ожидаемых результатов; кроме того, при случайном выборе тестовых данных могут оказаться непроверенными многие ситуации;
- **усложнение тестовых данных должно происходить постепенно.**

**Пример.** Система тестов для задачи нахождения корней квадратного уравнения  $ax^2 + bx + c = 0$  :

Номер теста	Проверяемый случай	Коэффициенты			Результаты
		a	b	c	

1	$d > 0$	1	1	-2	$x_1 = 1, x_2 = -2$
2	$d = 0$	1	2	1	Корни равны: $x_1 = -1, x_2 = -1$
3	$d < 0$	2	1	2	Действительных корней нет
4	$a=0, b=0, c=0$	0	0	0	Все коэффициенты равны нулю. $x$ — любое число.
5	$a=0, b=0, c \neq 0$	0	0	2	Неправильное уравнение
6	$a=0, b \neq 0$	0	2	1	Линейное уравнение. Один корень: $x = -0,5$
7	$a \neq 0, b \neq 0, c = 0$	2	1	0	$x_1 = 0, x_2 = -0,5$

## 8.9. Из каких этапов состоит процесс тестирования?

Процесс тестирования можно разделить на три этапа.

**1. Проверка в нормальных условиях.** Предполагает тестирование на основе данных, которые характерны для реальных условий функционирования программы.

**2. Проверка в экстремальных условиях.** Тестовые данные включают **граничные значения** области изменения входных переменных, которые должны восприниматься программой как правильные данные. Типичными примерами таких значений являются очень маленькие или очень большие числа и отсутствие данных. Еще один тип экстремальных условий — это **граничные объемы** данных, когда массивы состоят из слишком малого или слишком большого числа элементов.

**3. Проверка в исключительных ситуациях.** Проводится с использованием данных, значения которых лежат **за пределами допустимой области изменений**. Известно, что все **программы разрабатываются в расчете на обработку какого-то ограниченного набора данных**. Поэтому важно получить ответ на следующие вопросы:

- что произойдет, если программе, не рассчитанной на обработку отрицательных и нулевых значений переменных, в результате какой-либо ошибки придется иметь дело как раз с такими данными?
- как будет вести себя программа, работающая с массивами, если количество их элементов превысит величину, указанную в объявлении массива?
- что произойдет, если числа будут слишком малыми или слишком большими?

Наихудшая ситуация складывается тогда, когда **программа воспринимает неверные данные как правильные и выдает неверный, но правдоподобный результат**.

Программа должна сама отвергать любые данные, которые она не в состоянии обрабатывать правильно.



## 8.10. Каковы характерные ошибки программирования?

Ошибки могут быть допущены на всех этапах решения задачи — от ее постановки до оформления. Разновидности ошибок и соответствующие примеры приведены в таблице:

Вид ошибки	Пример
Неправильная постановка задачи	Правильное решение неверно сформулированной задачи
Неверный алгоритм	Выбор алгоритма, приводящего к неточному или эффективному решению задачи
Ошибка анализа	Неполный учет ситуаций, которые могут возникнуть; логические ошибки
Семантические ошибки	Непонимание порядка выполнения оператора
Синтаксические ошибки	Нарушение правил, определяемых языком программирования
Ошибки при выполнении операций	Слишком большое число, деление на ноль, извлечение квадратного корня из отрицательного числа и т. п.
Ошибки в данных	Неудачное определение возможного диапазона изменения данных
Опечатки	Перепутаны близкие по написанию символы, например, цифра 1 и буквы <i>I</i> , <i>l</i>
Ошибки ввода-вывода	Неверное считывание входных данных, неверное задание форматов данных

## 8.11. Является ли отсутствие синтаксических ошибок свидетельством правильности программы?

Обычно синтаксические ошибки выявляются на этапе трансляции. Многие же другие ошибки транслятору выявить невозможно, так как **транслятору неизвестны замыслы программиста**.

**Отсутствие сообщений машины о синтаксических ошибках является необходимым, но не достаточным условием, чтобы считать программу правильной.**

Примеры синтаксических ошибок:

- пропуск знака пунктуации;
- несогласованность скобок;
- неправильное формирование оператора;
- неверное образование имен переменных;
- неверное написание служебных слов;



- отсутствие условий окончания цикла;
- отсутствие описания массива и т.п.

## 8.12. Какие ошибки не обнаруживаются транслятором?

Существует множество ошибок, которые транслятор выявить не в состоянии, если используемые в программе операторы сформированы верно. Приведем примеры таких ошибок.

### Логические ошибки:

- неверное указание ветви алгоритма после проверки некоторого условия;
- неполный учет возможных условий;
- пропуск в программе одного или более блоков алгоритма.

### Ошибки в циклах:

- неправильное указание начала цикла;
- неправильное указание условий окончания цикла;
- неправильное указание числа повторений цикла;
- бесконечный цикл.

### Ошибки ввода-вывода; ошибки при работе с данными:

- неправильное задание тип данных;
- организация считывания меньшего или большего объема данных, чем требуется;
- неправильное редактирование данных.

### Ошибки в использовании переменных:

- использование переменных без указания их начальных значений;
- ошибочное указание одной переменной вместо другой.

### Ошибки при работе с массивами:

- массивы предварительно не обнулены;
- массивы неправильно описаны;
- индексы следуют в неправильном порядке.

### Ошибки в арифметических операциях:

- неверное указание типа переменной (например, целочисленного вместо вещественного);
- неверное определение порядка действий;
- деление на нуль;
- извлечение квадратного корня из отрицательного числа;
- потеря значащих разрядов числа.

Все эти ошибки обнаруживаются с помощью тестирования.

## 8.13. В чем заключается сопровождение программы?

**Сопровождение программ — это работы, связанные с обслуживанием программ в процессе их эксплуатации.**

Многократное использование разработанной программы для решения различных задач заданного класса требует проведения следующих дополнительных работ:

- исправление обнаруженных ошибок;
- модификация программы для удовлетворения изменяющихся эксплуатационных требований;
- доработка программы для решения конкретных задач;
- проведение дополнительных тестовых просчетов;
- внесение исправлений в рабочую документацию;
- усовершенствование программы и т.д.

Применительно ко многим программам работы по сопровождению поглощают более половины затрат, приходящихся на весь период времени существования программы (начиная от выработки первоначальной концепции и кончая моральным ее устареванием) в стоимостном выражении.

Программа, предназначенная для длительной эксплуатации, должна иметь соответствующую документацию и инструкцию по её использованию.

## 8.14. Вопросы для самоконтроля

- 8.1. Какие основные этапы включает в себя решение задач на компьютере?
- 8.2. Какие этапы компьютерного решения задач осуществляются без участия компьютера?
- 8.3. Что называют математической моделью объекта или явления?
- 8.4. Почему невозможно точное исследование поведения объектов или явлений?
- 8.5. Какие способы моделирования осуществляются с помощью компьютера?
- 8.6. Из каких последовательных действий состоит процесс разработки программы?
- 8.7. Доказывает ли получение правдоподобного результата правильность программы?
- 8.8. Какие ошибки могут остаться невыявленными, если не провести проверку (просмотр, прокрутку) программы?
- 8.9. Чем тестирование программы отличается от её отладки?
- 8.10. Каким образом программа-отладчик помогает исследовать поведение программы в процессе её выполнения?

- 8.11. Как следует планировать процесс отладки программы?
- 8.12. Можно ли с помощью тестирования доказать правильность программы?
- 8.13. На какой стадии работы над программой вычисляются эталонные результаты тестов?
- 8.14. Назовите основные этапы процесса тестирования.
- 8.15. В чём заключается отличие синтаксических ошибок от семантических?
- 8.16. О чём свидетельствует отсутствие сообщений машины о синтаксических ошибках?
- 8.17. Какие разновидности ошибок транслятор не в состоянии обнаружить?
- 8.18. Для чего программам требуется сопровождение?

---

## 8.15. Упражнения

**Составьте системы тестов для решения следующих задач:**

- 8.1. Найдите наибольший общий делитель двух заданных целых чисел.
- 8.2. Найдите наименьшее общее кратное двух заданных целых чисел.
- 8.3. Определите, является ли заданное число нечетным двузначным числом.
- 8.4. Заданы площади квадрата и круга. Определите, поместится ли квадрат в круге.
- 8.5. Решите биквадратное уравнение.
- 8.6. Найдите среднее арифметическое положительных элементов заданного одномерного массива.
- 8.7. Элементы заданного одномерного массива разделите на его первый элемент.
- 8.8. Определите, лежит ли заданная точка на одной из сторон треугольника, заданного координатами своих вершин.
- 8.9. Определите, имеют ли общие точки две плоские фигуры — треугольник с заданными координатами его вершин и круг заданного радиуса с центром в начале координат.
- 8.10. Задано целое  $A > 1$ . Найдите наименьшее целое неотрицательное  $k$ , при котором  $2^k > A$ .
- 8.11. Дана последовательность целых чисел. Определите, со скольких чётных чисел она начинается.
- 8.12. В заданном двумерном массиве найдите количество строк, не содержащих нули.

**8.13.** Определите, сколько строк заданного двумерного массива содержат элементы из заданного диапазона.

**8.14.** Преобразуйте число, заданное в римской системе счисления, в число десятичной системы.

---

---