ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ЭТАПЫ РЕШЕНИЯ ЗАДАЧ НА ЭВМ

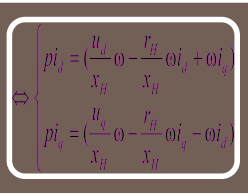
О.Ю. Лягинова, М.Г. Можаева
кафедра математики и информатики ЧГУ

Литература

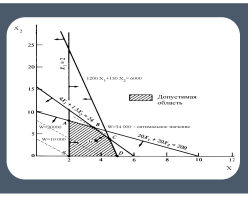
1. Немнюгин С.А. Turbo Pascal : Программирование на языке высокого уровня : учебник для вузов / С. А. Немнюгин - 2-е изд. - СПб. : Питер, 2007. - 543 с.
2. Крылов Е.В. Техника разработки программ : учебник для вузов : В 2-х книгах. Кн.1 : Программирование на языке высокого уровня / Е. В. Крылов, В. А. Острейковский, Н. Г. Типикин ; Крылов Е.В., Острейковский В.А., Типикин Н.Г. - Москва : Высшая школа, 2007. - 376 с.
3. Парфилова Н.И. Программирование. Структурирование программ и данных : учебник для студ. учреждений высш. проф. образования / Н.И. Парфилова, А.Н. Пылькин, Б.Г. Трусов ; под ред. Б.Г. Трусова. — М. : Издательский центр «Академия», 2012. — 240 с. — (Сер. Бакалавриат).
4. http://ru.wikibooks.org/wiki/Основы_функционального_программирования/Вводная_лекция

Этапы решения задачи с помощью средств ВТ

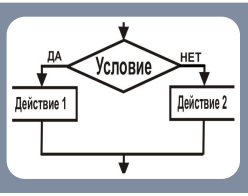
1. Разработка технического задания (постановка задачи на содержательном уровне)



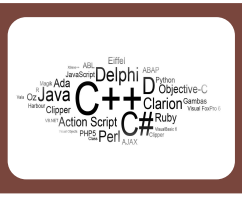
2. Формализация задачи (построение математической модели)



3. Выбор (разработка) метода решения задачи



4. Разработка алгоритма (алгоритмизация)



5. Выбор языка/системы программирования

Этапы решения задачи с помощью средств ВТ

6. Разработка программы (программирование)

7. Отладка и тестирование программы

8. Оптимизация программы

9. Документирование программы

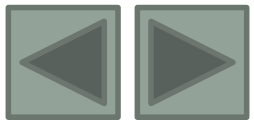
10. Вычисление и обработка результатов

1. Постановка задачи

- Содержательный анализ существа задачи.
- Изучение общих свойств рассматриваемого физического явления или объекта.
- Определение конечной цели и результатов работы.
- Анализ известной информации и определение исходных данных.
- Выработка общего подхода к исследуемой проблеме (выяснение, существует ли решение поставленной задачи и единственно ли оно);
- Анализ возможностей используемой вычислительной среды.

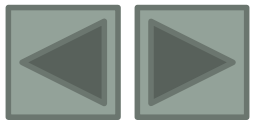
2. Формализация задачи

- Формализация задачи сводится к построению *математической модели* рассматриваемого объекта, явления или процесса, когда в результате предыдущего анализа существа решаемой задачи устанавливается её принадлежность к одному из известных классов задач и выбирается соответствующий математический аппарат, определяется формат исходных данных и результатов работы, вводится определенная система условных обозначений.



2. Формализация задачи

Математическая модель —
это система математических соотношений,
учитывающих наиболее существенные взаимосвязи
в изучаемом классе объектов/явлений и их
свойства,
в совокупности с определенной областью
допустимых значений исходных данных и
областью допустимых значений искомых
результатов.



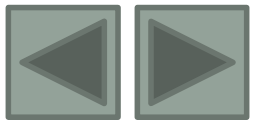
3. Выбор метода решения задачи

После математической постановки задачи отвлекаются от её предметной сущности и оперируют с абстрактными математическими понятиями, величинами, формулами.

3. Выбор метода решения задачи

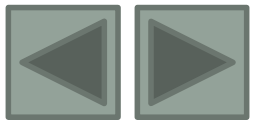
При выборе метода надо учитывать:

1. характеристики самого метода, сложность формул и соотношений, связанных с этим методом;
2. необходимую точность вычислений.



4. Разработка алгоритма

- **Процесс алгоритмизации** — это отдельный этап, которому придаётся особая значимость.
- Именно на этом этапе осуществляется процесс разработки структуры алгоритма, реализующего выбранный метод решения, и осуществляется запись «придуманной» структуры на языке, «понятном» самому разработчику.



4. Разработка алгоритма

- Можно использовать различные способы описания алгоритмов, отличающиеся по простоте и наглядности.
- В практике программирования наибольшее распространение получили:
  1) словесная запись алгоритмов;
  2) схемы алгоритмов (блок-схемы);
  3) структурограммы (диаграммы Насси — Шнейдермана).

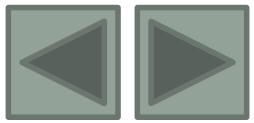
4. Разработка алгоритма

Построение алгоритма включает:

- разложение вычислительного процесса решения задачи на возможные составные части;
- установление порядка их следования;
- описание содержания каждой такой части в той или иной форме;
- последующей проверке, которая должна показать, обеспечивается ли реализация выбранного метода.

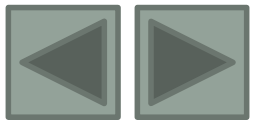
4. Разработка алгоритма

- В процессе разработки алгоритм проходит несколько **этапов детализации**.
- Первоначально составляется **укрупненная схема алгоритма**, в которой отражаются наиболее важные и существенные связи между исследуемыми процессами (или частями процесса).
- Ориентируясь на крупноблочную структуру алгоритма, можно быстрее и проще разработать несколько различных его вариантов провести их анализ, оценку и выбрать наилучший (оптимальный).



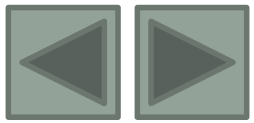
4. Разработка алгоритма

- Каждый элемент крупноблочной схемы алгоритма должен быть максимально самостоятельным и логически завершённым в такой степени, чтобы дальнейшую его детализацию можно было выполнять независимо от детализации остальных элементов.

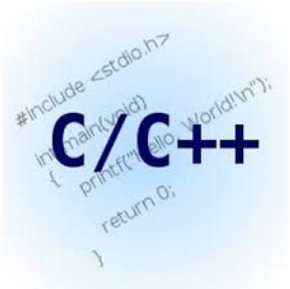


4. Разработка алгоритма

- На последующих этапах **детализируются** выделенные на предыдущих этапах части вычислительного процесса, имеющие некоторое самостоятельное значение.
- На каждом этапе детализации выполняется многократная проверка и исправление схемы алгоритма.
- Подобный подход позволяет во многом избежать возможных ошибочных решений.



5. Выбор языка/системы программирования



Язык программирования — язык, предназначенный для представления программ (ГОСТ 28397-89)

Низкого уровня
учитывает архитектуру процессора, например, Ассемблер

Высокого уровня
не учитывает архитектуру процессора, например: Паскаль; Бейсик; Си

5. Выбор языка/системы программирования

- **Система программирования** – комплекс средств, предназначенных для создания и эксплуатации программ на конкретном языке программирования на ЭВМ определенного типа.
- Примеры: Turbo Pascal; Pascal ABC; Delphi; C++ Builder и др.
- Системы программирования включают в себя:
  - текстовый редактор;
  - транслятор;
  - набор библиотек;
  - отладчик и др.

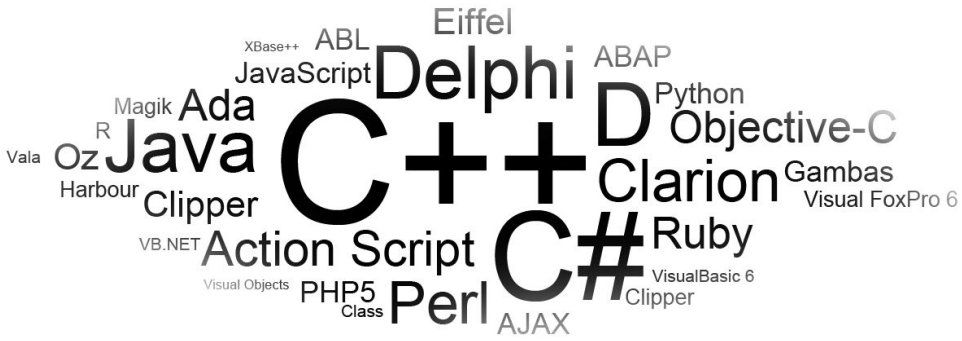
5. Выбор языка/системы программирования

- Примером *интегрированной среды разработки* программного обеспечения, предназначенной для написания консольных приложений, приложений с графическим интерфейсом пользователя, веб-сайтов, веб-приложений, веб-служб и др., является *Microsoft Visual Studio*.
- Microsoft Visual Studio включает в себя один или несколько компонентов: *Visual Basic .Net; Visual C++; Visual C#; Visual F#* и др. Это позволяет разработчику выбрать наиболее удобную систему программирования для реализации определённого этапа работы.

5. Выбор языка/системы программирования

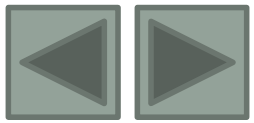
Выбор языка/системы программирования определяется следующими факторами:

- типом решаемой задачи;
- располагаемыми вычислительными средствами;
- вкусами и знаниями заказчика и разработчика.



6. Разработка программы

Процесс программирования –
(в широком смысле) процесс создания программ;
(в узком смысле) процесс кодирования на одном из языков программирования.



6. Разработка программы

- Программа, написанная на языке высокого уровня (исходный код), проходит этап **трансляции** – преобразования в машинный код.

6. Разработка программы

компилятор **компоновщик**

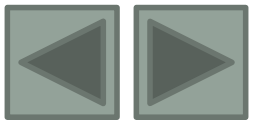
промежуточное состояние программы в относительных адресах с неразрешёнными внешними ссылками и использованием всей логической структуры программы

абсолютный/ загрузочный код с абсолютной адресацией машинных команд, может быть сохранен в exe-файле и выполнен

Загрузка программы из exe-файла в память машины для её выполнения осуществляется служебной программой – **загрузчиком**.

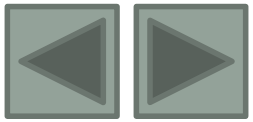
6. Разработка программы

- **Интерпретатор** сразу производит анализ, перевод в машинный код и выполнение программы строка за строкой.
- Поэтому интерпретатор должен находиться в оперативной памяти в течение всего времени выполнения программы пользователя.
- При интерпретации скорость выполнения программы существенно снижается, однако весь процесс прохождения программы на ЭВМ упрощается и имеется возможность организации диалогового (интерактивного) режима отладки и выполнения программы.



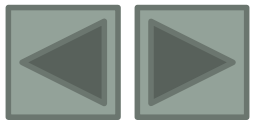
7. Отладка и тестирование программы

- **Отладка программы** – это процесс поиска и устранения ошибок.
- Часть ошибок формального характера, связанных с нарушением правил записи конструкций языка или отсутствием необходимых описаний, обнаруживает транслятор, производя синтаксический анализ текста программы. Транслятор выявляет ошибки и сообщает о них, указывая их тип и место в программе. Такие ошибки называются **синтаксическими**.



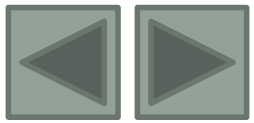
7. Отладка и тестирование программы

- Ошибочные ситуации могут возникнуть и при выполнении программы, например деление на ноль, извлечение корня квадратного из отрицательного числа, попытка открыть несуществующий файл.
- Такие ошибки называются **семантическими**, они связаны с неправильным содержанием действий или использованием недопустимых значений величин.



7. Отладка и тестирование программы

- Программа, не имеющая синтаксических и семантических ошибок, может не дать верных результатов из-за **логических** ошибок в алгоритме.
- Ошибки подобного рода могут возникнуть на этапе постановки задачи, разработки математической модели, разработки алгоритма.



7. Отладка и тестирование программы

Рассмотрим программу вычисления значения функции $f(x)$ при целом x.

$$f(x)=\begin{cases} 3|x|, & x<-2 \\ 9/x, & -2\le x\le 2,\ x\neq 0 \\ 12, & x=0 \\ \sin x, & x>2 \end{cases}$$

```
var x: integer; f: real;
begin
writeln ('Введите целое x);
readln (x);
if x<-2 thn f:=3*abs(x) else
  if (x>=-2) and (x<=2) then f:=9/x else
    f:=sin(x);
writeln ('f(',x,')=',f:6:2);
end.
```

Синтаксическая ошибка

writeln ('Введите целое x')
then

деление на 0 при x=0

Семантическая ошибка

Логическая ошибка

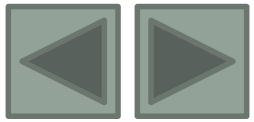
7. Отладка и тестирование программы

Тестирование программы — это выполнение программы на наборах исходных данных (тестах), для которых известны результаты, полученные другим методом.

7. Отладка и тестирование программы

7. Отладка и тестирование программы

При тестировании программы простой и действенный метод дополнительного контроля над ходом её выполнения — **получение контрольных точек**, т.е. контрольный вывод промежуточных результатов.



8. Оптимизация программы

Оптимизация программы – это её модификация с целью повышения эффективности работы.

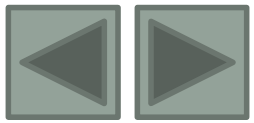
- Для оптимизации требуется найти критическую часть кода, которая является основным потребителем необходимого ресурса.
- Для поиска узких мест используются специальные программы — **профайлеры**.

9. Документирование программы

- Главная цель документации состоит в том, чтобы помочь стороннему пользователю понять и использовать программу.

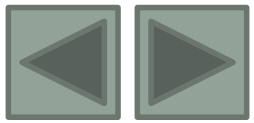
9. Документирование программы

- **Внешняя документация** представляет собой сведения о программе, не содержащиеся в самой программе.
- В зависимости от размеров и сложности программы внешняя документация может принимать различные формы:
  - схемы или словесные описания алгоритмов;
  - инструкции для пользователей;
  - образцы входных и выходных данных;
  - полное описание процесса построения программы;
  - ссылки на источники информации и др.



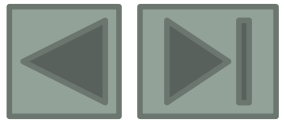
9. Документирование программы

- **Программная документация** реализуется с помощью комментариев (в начале программы – в виде заголовка и вводных комментариев; поясняющих — внутри текста программы), а также рационального выбора имён, применения стандартных методов структурирования программ.

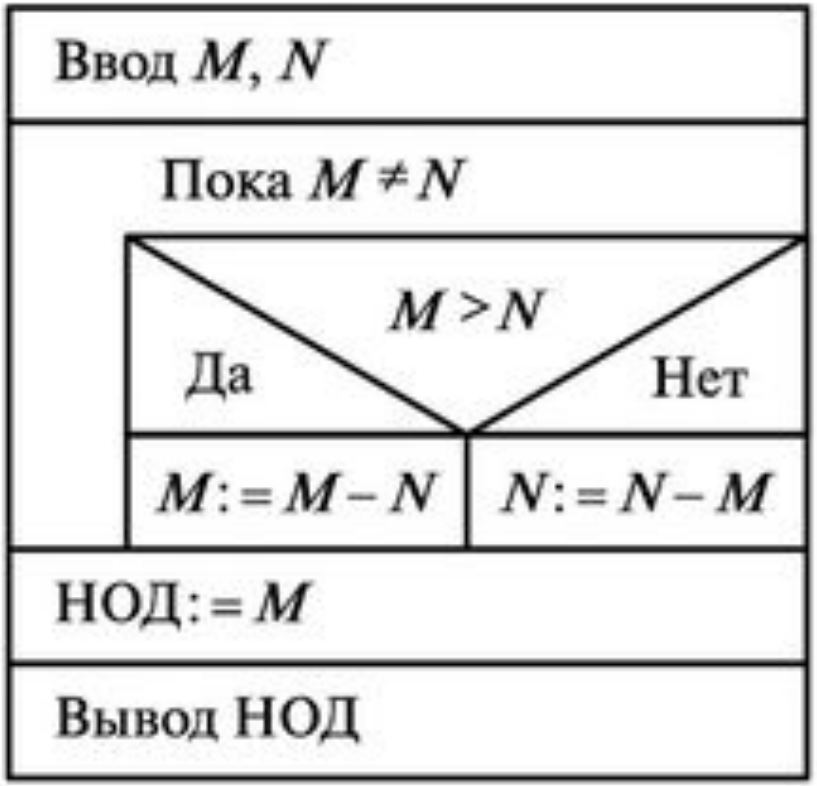


10. Вычисления и обработка результатов

- Только после того как появится уверенность, что программа обеспечивает получение правильных результатов, можно приступать непосредственно к расчётам.
- После завершения расчётов полученные результаты используются в практической деятельности.



Структурограмма алгоритма Евклида



ЧЕРЕПОВЕЦКИЙ
ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ

Спасибо за внимание!

Выход