Среда программирования

Прогресс технологии дает нам все более совершенные средства для движения вспять.

О. Хаксли

Содержание

- Понятие среды программирования
- Техника разработки программ
- Классификация ошибок в программе
- Использование отладчика

Среда программирования

- *Среда (система) программирования* – совокупность инструментов, обеспечивающих преобразование программы на некотором языке программирования в выполнимые вычисления.

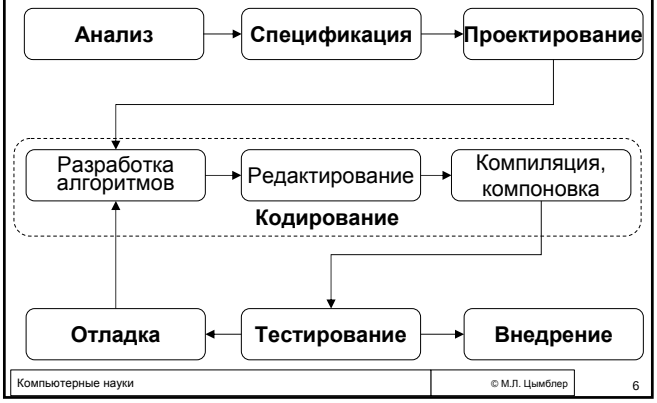
Основные компоненты среды

- *Редактор* – средство создания и изменения *исходных* файлов с текстом программы.
- *Компилятор* – транслирует исходный файл в *объектный* файл, содержащий команды в машинном коде для конкретного компьютера.
- *Компоновщик (редактор связей)* – собирает объектные файлы программы и формирует *исполняемый* файл (*разрешая* внешние ссылки между объектными файлами).
- *Отладчик* – средство управления выполнением исполняемого файла на уровне отдельных операторов программы для диагностики ошибок.

Прочие компоненты среды

- *Библиотекарь* – средство ведения совокупностей объектных файлов (*библиотек*).
- *Профилировщик* – средство измерения времени выполнения программных компонент для последующей оптимизации критических компонент.
- *Загрузчик* – копирует исполняемый файл с диска в память и осуществляет его запуск.
- *Средство версионирования* – регистрация всех изменений исходного текста с возможностью отката.

Цикл разработки программы

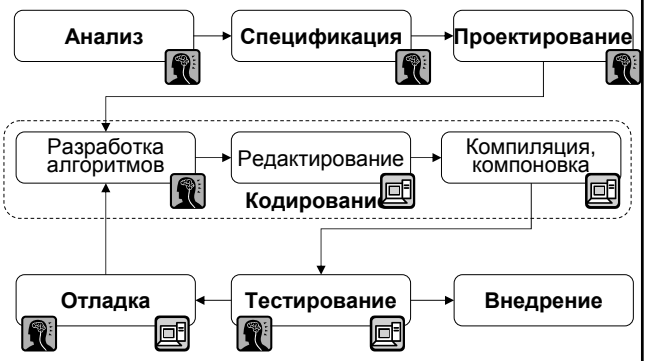


Цикл разработки программы

- *Анализ* – установить, *что* должна делать программа (но не *как* она это должна делать).
- *Спецификация* – записать требования к программе в формальном виде.
- *Проектирование* – разработать структуру программы.
- *Кодирование* = разработка алгоритмов + редактирование + компиляция + компоновка.
- *Тестирование* – подготовить эталонные входные и соответствующие выходные данные (*тесты*), запустить программу и сравнить полученные данные с эталонными.
- *Отладка* – найти и исправить ошибки.

Компьютерные науки © М.Л. Цымблер 7

Цикл разработки программы



Компьютерные науки © М.Л. Цымблер 8

Пример разработки программы

- Задача
  Если треугольник с заданными сторонами существует, то определить его вид.
- Анализ
  - **Формулировка**
    Вид треугольника – равносторонний, равнобедренный, общий или …?
    Длины сторон – целые или вещественные числа? в каком диапазоне?
  - **Входные данные**
    Откуда считываются данные – с клавиатуры, из (какого?) файла или …? Возможен ли ввод некорректных данных (строка символов вместо числа)?
  - **Выходные данные**
    Куда записываются – на экран и/или в (какой?) файл или …? В каком виде?

Компьютерные науки © М.Л. Цымблер 9

Пример разработки: Спецификация

- **Формулировка**
  Заданы длины сторон треугольника. Если треугольник с такими сторонами существует, определить его вид: равносторонний, равнобедренный, общий. Иначе сообщить, что треугольник не существует.
- **Входные данные**
  Три вещественных числа (не обязательно положительные!). Считываются из текстового файла INPUT.TXT. Всегда корректны.
- **Выходные данные**
  Записываются в текстовый файл OUTPUT.TXT. Одно из сообщений: РАВНОСТОРОННИЙ, РАВНОБЕДРЕННЫЙ, ОБЩЕГО ВИДА, НЕ СУЩЕСТВУЕТ
- **Примеры входных и выходных данных**

INPUT.TXT	OUTPUT.TXT
1 1 1	РАВНОСТОРОННИЙ
1 2 3	НЕ СУЩЕСТВУЕТ
3 4 5	ОБЩЕГО ВИДА

Компьютерные науки © М.Л. Цымблер 10

Пример разработки: Проектирование

Задача о треугольнике

- Вид (заведомо существующего) треугольника
- Проверка существования треугольника

ВИД

Вход вещ *a*, *b*, *c*
Выход цел *результат*
Комментарий
Возвращает вид треугольника с длинами сторон a, b, c:
3 - равносторонний
2 - равнобедренный
1 - иначе.
Треугольник заведомо должен существовать.

ПРОВЕРКА

Вход вещ *a*, *b*, *c*
Выход лог *результат*
Комментарий
Возвращает Истину, если существует треугольник с длинами сторон a, b, c.
Иначе возвращает Ложь.

Компьютерные науки © М.Л. Цымблер 11

Пример разработки: Кодирование

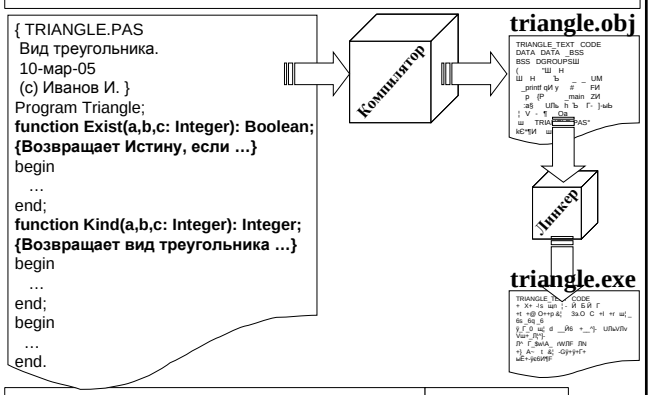
```
function Exist(a,b,c: Integer): Boolean;
{Возвращает Истину, если …}
begin
  …
end;
```

```
function Kind(a,b,c: Integer): Integer;
{Возвращает вид треугольника …}
begin
  …
end;
```

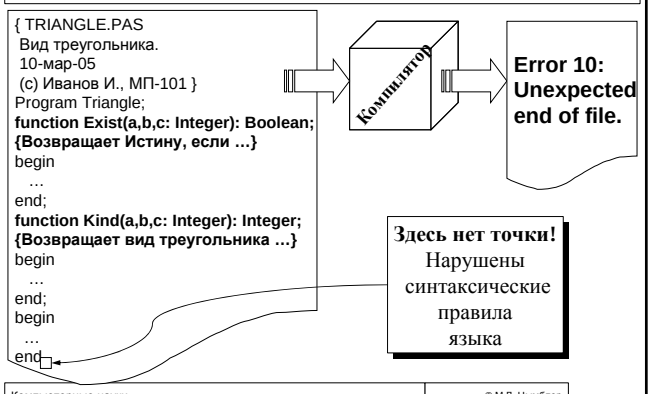
```
{ TRIANGLE.PAS
 Вид треугольника.
 10-мар-05
 (c) Иванов И. }
Program Triangle;
function Exist(a,b,c: Integer): Boolean;
{Возвращает Истину, если …}
begin
  …
end;
function Kind(a,b,c: Integer): Integer;
{Возвращает вид треугольника …}
begin
  …
end;
begin
  …
end.
```

Компьютерные науки © М.Л. Цымблер 12

Пример разработки: Компиляция, компоновка



Синтаксическая ошибка

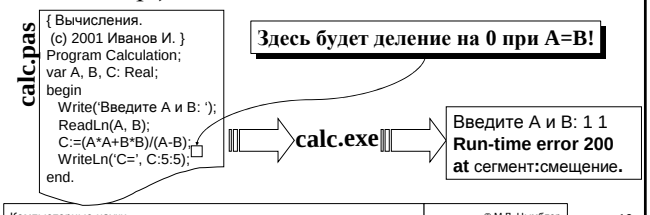


Пример разработки: Тестирование

№	Входные данные			Ожидаемый результат	Действительный результат	Тест пройден?
	a	*b*	*c*			
1	1	1	1	РАВНОСТОРОННИЙ	РАВНОСТОРОННИЙ	√
2	1	2	3	НЕ СУЩЕСТВУЕТ	ОБЩЕГО ВИДА	×
3	3	4	5	ОБЩЕГО ВИДА	ОБЩЕГО ВИДА	√
4	0	0	0	НЕ СУЩЕСТВУЕТ	РАВНОСТОРОННИЙ	×
5	4	4	5	РАВНОБЕДРЕННЫЙ	РАВНОБЕДРЕННЫЙ	√
6	2	1	1	НЕ СУЩЕСТВУЕТ	РАВНОБЕДРЕННЫЙ	×
7	-1	-1	-1	НЕ СУЩЕСТВУЕТ	Runtime error 106 at 0BEB:0026.	×

Ошибка времени выполнения

- *Ошибка времени выполнения (run-time error)* происходит при выполнении (синтаксически верной) программы, когда она производит какое-либо недопустимое действие (деление на ноль и др.).

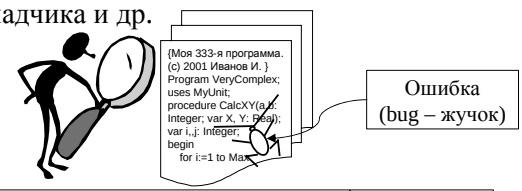


Логическая ошибка

- *Логическая (алгоритмическая) ошибка* – ошибка разработки и написания алгоритма. Программа не содержит ни синтаксических ошибок, ни ошибок времени выполнения, но делает не то, что хотел автор программы.

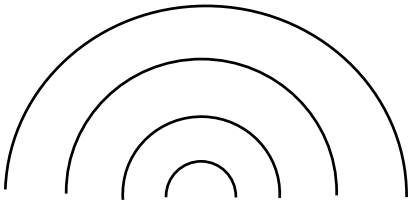
Отладка

- *Отладка (Debugging)* – процесс локализации и исправления ошибок, выявленных во время тестирования программы.
- *«Сухая» отладка* – по листингу (тексту) программы, без использования компьютера, отладчика и др.



Отладчик

- *Отладчик (Debugger)* – программа, позволяющая отслеживать процесс выполнения программы по ее исходному тексту и просматривать текущие значения переменных.



myprog.exe

Отладчик

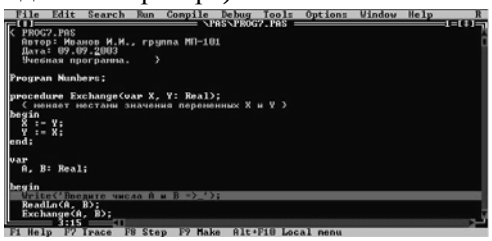
Среда разработки

Операционная система

Компьютерные науки © М.Л. Цымблер 19

Основные функции отладчика

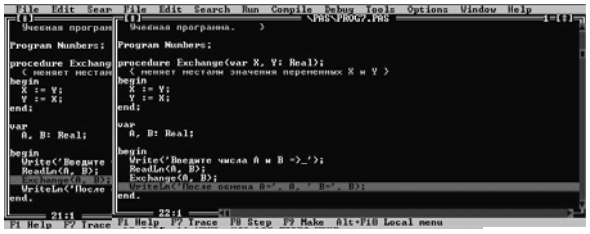
- *Трассировка* – пошаговое выполнение программы. Шагу соответствует *одна строка исходного текста* (в которой может быть более одного оператора).



Компьютерные науки © М.Л. Цымблер 20

Основные функции отладчика

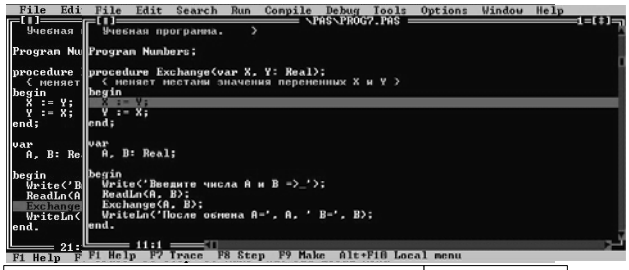
- *Режим "без трассы подпрограмм"* – пошаговое выполнение программы, при вызов подпрограммы отрабатывается как один оператор.



Компьютерные науки © М.Л. Цымблер 21

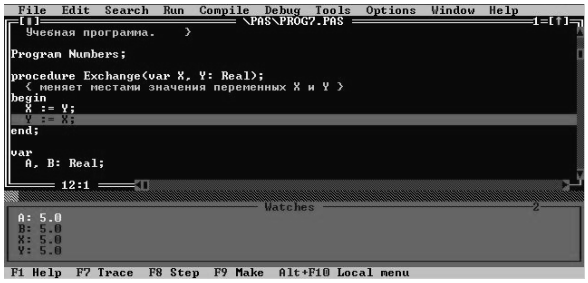
Основные функции отладчика

- *Режим "трасса подпрограмм"* – пошаговое выполнение программы, при котором трасса включает все операторы подпрограмм.



Основные функции отладчика

- *Просмотр значений переменных* при пошаговом выполнении программы.



Основные функции отладчика

- *Точка останова (breakpoint)* приостанавливает выполнение программы. Может быть установлена *только на выполняемом операторе* (не на комментарии и др.). С точкой останова может быть связано логическое условие ее включения.