Алгоритмы и алгоритмические языки

Лекция 17

Программирование "сверху вниз". Тестирование. Отладка

Программирование сверху вниз

(нисходящее проектирование, пошаговая детализация)

Цели:

- Сделать процесс разработки программ целенаправленным, систематичным,
- Сократить время и усилия, затрачиваемые на разработку программ

Основная идея:

На каждом шаге сводить задачи к более простым подзадачам, чтобы получить такие подзадачи, решение которых очевидно; затем объединить решение подзадач в единую программу.

Программирование сверху вниз. Пример

Задача:

Найти решение системы линейных уравнений n-го порядка (n=20)

$$a_{11}*x_1 + a_{12}*x_2 + \ldots + a_{1n}*x_n = b_1$$
$$a_{21}*x_1 + a_{22}*x_2 + \ldots + a_{2n}*x_n = b_2$$
$$\ldots$$
$$a_{n1}*x_1 + a_{n2}*x_2 + \ldots + a_{nn}*x_n = b_n$$

или Ax=b (в векторном виде)

Дополнительное условие: det A<>0

==> система имеет решение и притом единственное

Программирование сверху вниз. Пример

Выбираем метод решения

метод Гаусса

$$\begin{cases} 2x_1 - x_2 + x_3 = 3 \quad |*3 \quad |*1 \\ 3x_1 + 2x_2 - x_3 = 4 \quad |*2 \\ x_1 + x_2 + x_3 = 6 \quad\quad |*2 \end{cases}$$

Приводим систему к треугольному виду:

Исключаем x1 из всех уравнений, кроме первого

Программирование сверху вниз. Пример

Выбираем метод решения

метод Гаусса

$$\begin{cases} 2x_1 - x_2 + x_3 = 3 \quad |*3 \quad |*1 \\ 3x_1 + 2x_2 - x_3 = 4 \quad |*2 \\ x_1 + x_2 + x_3 = 6 \quad\quad |*2 \end{cases} \qquad \begin{cases} 2x_1 - x_2 + x_3 = 3 \\ 7x_2 - 5x_3 = -1 \quad |*3 \\ 3x_2 + x_3 = 9 \quad\quad |*7 \end{cases}$$

Приводим систему к треугольному виду:

Исключаем x1 из всех уравнений, кроме первого
Исключаем x2 из всех уравнений, кроме первого и второго
и т.д.

Программирование сверху вниз. Пример

Выбираем метод решения

метод Гаусса

$$\begin{cases} 2x_1 - x_2 + x_3 = 3 & |*3 \quad |*1 \\ 3x_1 + 2x_2 - x_3 = 4 & |*2 \\ x_1 + x_2 + x_3 = 6 & |*2 \end{cases} \quad \begin{cases} 2x_1 - x_2 + x_3 = 3 \\ 7x_2 - 5x_3 = -1 & |*3 \\ 3x_2 + x_3 = 9 & |*7 \end{cases} \quad \begin{cases} 2x_1 - x_2 + x_3 = 3 \\ 7x_2 - 5x_3 = -1 \\ 22x_3 = 66 \end{cases}$$

Приводим систему к треугольному виду:

Исключаем x1 из всех уравнений, кроме первого
Исключаем x2 из всех уравнений, кроме первого и второго
и т.д.

Решение системы с треугольной матрицей

Программирование сверху вниз. Пример

$$\begin{cases} 2x_1 - x_2 + x_3 = 3 \\ 7x_2 - 5x_3 = -1 \\ 22x_3 = 66 \end{cases}$$

Решение: (1,2,3)

Замечания:

1. В общем случае решение – вектор вещественных чисел
2. При выполнении очередного шага приведения к треугольной матрице можем получить нулевой диагональный элемент:

$$\begin{cases} x_2 + x_3 = -1 \\ 2x_1 - x_2 + 7x_3 = 5 \\ x_1 + 2x_2 - 3x_3 = 0 \end{cases} \Rightarrow \begin{cases} 2x_1 - x_2 + 7x_3 = 5 \\ x_2 + x_3 = -1 \\ x_1 + 2x_2 - 3x_3 = 0 \end{cases}$$

3. В программе работаем с матрицей и векторами

Программирование сверху вниз. Пример

1 шаг детализации

begin
ввод исходных данных;
приведение системы к треугольному виду;
решение треугольной системы;
вывод результатов
end

2 шаг детализации

```
begin
   {ввод исходных данных}
    ввод A;
    ввод b;
   {приведение системы к треугольному виду}
    for i:=1 to n-1 do
    begin
        if A[i,j]=0 then
        begin  поиск k-го уравнения с A[k,i] <> 0;
                  перестановка i-го и k-го уравнений
        end;
        for k:=i+1 to n do
              исключение x[i] из k-го уравнения
    end;
   {решение треугольной системы}
   for i:=n downto 1 do  вычисление x[i];
   {вывод результатов}
   вывод x
end
```

Способы проверки программ

1. Проверка "вручную"

возможна только для небольших программ

ОС Microsoft Windows 3.1 (1992 г.) - 3 млн. строк кода
Windows 98 – 18 млн. строк кода
Windows XP (2000 г.) - 40 млн. строк кода

Red Hat Linux 6.2 (2000 г.) - 20 млн. строк кода
Red Hat Linux 7.1 (2001 г.) - 30 млн. строк кода

2. Тестирование

3. Верификация

Тестирование программ

С помощью тестов можно обнаружить ошибки, но нельзя доказать, что их нет.

Пример программы:

```
read(x);
if x<>6.079 then y:=x*x else y:=1.2
writeln(y);
```

Цель тестирования – поиск ошибок в программе, а не показ того, что она работает правильно.

Стратегии тестирования

Стратегия черного ящика

при тестировании не видим текст программы, тесты составляются на основе спецификации (описания) программы

Стратегия белого ящика

при тестировании мы знаем текст программы, можем учитывать ее структуру.

Стратегия черного ящика

1) Метод эквивалентных классов

2) Метод анализа граничных значений

Стратегия белого ящика

1) Метод покрытия операторов

Набор тестов должен быть таким, чтобы каждый оператор хотя бы один раз выполнился

2) Метод покрытия переходов (условий)

3) Метод покрытия маршрутов

Отладка программ

1) Локализация ошибки (определение места ошибки)

   - отладочные печати
   - использование отладчика

2) Определение причины ошибки

3) Исправление ошибки