МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2006
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2006
Proceedings of the Thirty Fifth Spring Conference of
the Union of Bulgarian Mathematicians
Borovets, April 5–8, 2006

МЕТОДОЛОГИЯ НА ИЗБИРАТЕЛНОТО ПОВЕДЕНИЕ В ПРОЦЕДУРНОТО ПРОГРАМИРАНЕ

Димитър Д. Добрев

Статията засяга методологични проблеми на процедурното програмиране. Изследвано е избирателното поведение на програмите (двузначно и многозначно) в различните му проявления, както и разнообразието от изразни средства в езика Си, позволяващи неговата програмна реализация. Върху примера на атрактивна етюдна задача (частен случай на задачата за многопараметричната класификация) са демонстрирани различни взаимозаменяеми подходи и техники за постигане на операторна, индексна и изчислителна избирателност.

1. Въведение. Курсовете по "Увод в програмирането" – училищни, университетски или професионални – традиционно следват структурата на избрания програмен език. Който и учебник по процедурно програмиране да разгърнем, неизменно ще открием следната тематична последователност: *обща структура на програмата, прости величини и типове, изрази, оператори за присвояване, оператори за вход и за изход, оператори за преходи и за цикли, съставни типове, функции, рекурсия и т.н.* Този методологичен подход, добре издържан от гледна точка на систематиката на езиковите средства, не бива обаче да бъде абсолютизиран. Естествено, право на съществуване имат и други възможни подходи. Въпрос на целесъобразност е кой и кога да предпочетем.

Без претенцията да предлага една изцяло нова структура в изграждането на учебното съдържание по процедурно програмиране, настоящата статия е опит да бъде защитен и един друг подход, различен от традиционния. Мотивите за това са продиктувани от стремежа към едно по-дълбоко вникване в същността на програмирането, а реалната практическа цел – възможността съществено да обогатим използваните в програмирането подходи и техники.

Като конкретна материя, върху която бихме могли да обосновем и илюстрираме подобна методологическа позиция, е избрана проблематиката на средствата за двузначен и многозначен избор в езиците за процедурно програмиране, а като конкретен език – езикът Си. По същество се предлага сечение през езика под такъв зрителен ъгъл, който би ни позволил да разгледаме проблема за избирателността в неговата общност. В резултат става възможно да се доближим до такова равнище на знания и умения, което граничи с представите за един истински професионализъм.

2. Избирателното поведение и неговата същност. Отправна точка за нашия анализ е убеждението ни, че **избирателното поведение** е същностно свойство

на компютрите, в качеството им на абстрактни субекти, използвани като моделни среди или пък като елементи на виртуални системи. В същност именно способността им за избирателно поведение е онова, което превръща машините в субекти, а използваните от тях данни – в информация.

Върху обичайните в процедурното програмиране форми на избирателно поведение и съответните им езикови средства са приложими различни класификации:

По отношение на значността си те биват ***двузначни*** и ***многозначни***.

По отношение на предмета си – ***изчислителни, операторни*** и ***нформационни.***

Приведени понятийно към езика Си, това са следните езикови средства:

- Условни изрази
- Оператори за двузначен избор (в две разновидности)
- Опеатори за многозначен избор
- Оператори за циклични процеси (в три разновидности)
- Операции за индексна селекция (едно- и многопараметрична)
- Операции за указателна (косвена) селекция.

Богатството на езиците за програмиране, като моделно средство, е заложено в:

- Разнообразието от директно реализираните в езика алгоритмични и информационни структури,
- Възможността за тяхната комбинирана употреба,
- Възможността за тяхната взаимозаменяемост.

Това е и дълбокият смисъл, който Уирт е вложил в заглавието на широко известната си книга "Алгоритми + Структури от данни = Програми" [1]. Високата практическа стойност на тази "формула" се основава именно на споменатите възможности за "комбинирана употреба" и за "взаимозаменяемост". Така, изправени пред обективната сложност на една задача, отчитайки нейната същност, както и моделните възможности и технологичните удобства на наличните езикови средства, можем в значително по-широки граници да търсим оптималния баланс между алгоритмичните и информационните компоненти на конструираното от нас програмно решение. Впрочем, именно тази прагматична цел е фокусът, в който са съсредоточени усилията ни в тази публикация.

3. Една етюдна задача. За нуждите на необходимите програмни илюстрации ще формулираме една проста етюдна задача. Поднесена под атрактивната форма на игра за отгатване, задачата за "Отгатване на зарчето" по същество представлява частен случай на важната за практиката задача за "Многопараметричната класификация".

Входни данни за отгатващата програма са аритметичните свойства на числото, което се е паднало при хвърлянето на зарчето. Програмата ги узнава чрез инициран от нея диалог, под формата на отговори на следните три въпроса:

“ Числото голямо ли е ? ” (1, 2 и 3 се считат за малки, а 4, 5 и 6 – за големи),
“Числото дели ли се на 2 ? ” и “Числото дели ли се на 3 ? ”

Целта е програмата да отгатне конкретното числото, което се е паднало.

Следният програмен диалог осигурява необходимите за това входни данни, при което незначителните отклонения от стандарта на Си не променят същноста на разглежданите тук въпроси.

```
# include <iostream.h> //
void dialog2 (char *question, int *answer) {
   char yn ;
        do {cout « question ; cin » yn; }
        while (!( yn=='Y' || yn=='y' || yn=='N' || yn=='n' ));
        *answer = ( yn== 'Y' || yn== 'y' ); }
void main (void) {
   int b, d, t; //признаци за ''голямо'', ''делимостна 2'' и ''делимост на 3''
   int r;       //остатък по модул 3
   int Z;       //краен числов резултат
        dialog2 ('' Числото голямо ли е ? '', & b);
        dialog2 ('' Числото дели ли се на 2 ? '', & d);
        dialog2 ('' Числото дели ли се на 3 ? '', & t);
        ..................................... }
```

Освен този вариант на задачата за зарчето, който нека означим като $Z_{2\times2\times2}$, бихме могли да опитаме и варианта $Z_{2\times3}$, основан на диалога:

```
        dialog2 ('' Числото голямо ли е ? '', & b);
        dialog3 ('' Какъв е остатъкът от делението на 3 ?, & r).
```

Естествено, ще потрябва аналогична диалогова функция dialog3 , допускаща тризначен числов отговор, с възможни стойности 0, 1 и 2.

```
        void dialog3 (char *question, int *answer) {
        do {cout « question ; cin » *answer; }
        while (!(0<=answer && answer<=2 )); }
```

Още една, допълнителна задача (да я означим с $C_{2\times2\times2}$), свързана с варианта $Z_{2\times2\times2}$, е задачата за установяване на логическата непротиворечивост на тройката конкретни двузначни отговори, получени с помощта на функцията dialog2.

Дългогодишният ни преподавателски опит в Софийския университет ни е убедил, че като правило, при подобни задачи се търси решение в чисто операторен стил, с многократно и не винаги оправдано използване на оператори if. Обяснение на този феномен е явно заложената в структурните езици възможност за вложена употреба на условните оператори, както и близките до естествените езици лексика и синтаксис на съответните езикови конструкции. Подобно езиково сходство се оказва обаче доста подвеждащо, защото в естествените езици, тъкмо поради структурната им сложност, рядко се срещат алтернативни конструкции с дълбочина на вложение по-голяма от 2.

4. Конструиране на критерии за двузначен и многозначен избор. В своята професионална практика програмистите твърде често се сблъскват със задачи с усложнена логика. Естествено е при конструирането на съответните им алгоритми,

ръководен фактор да бъде двузначността или многозначността на избора, обективно свързан с решаването на съответната задача.

Като контрапункт на обичайно прилагания от програмистите операторен стил, с използване на вложени условни оператори, тук съзнателно ще търсим и други възможни подходи. В този смисъл, ключов за нас ще се окаже проблемът за конструиране на двузначни и многозначни селектори.

В случаите на задачи с двузначен избор, фундаментална от теоретична и практическа гледна точка е добре известната ***дизюнктивна нормална форма*** на Бул.

В случаите на задачи с многозначен избор, решение на въпроса откриваме във възможността за конструиране на нехомогенни (булево-числови) изрази.

Пристъпвайки към демонстрация на подобни подходи и техники, ще представим в табличен вид комбинаториката на първичните данни, свързани с вариантите на нашата задача, както и някои вторични величини, свързани с търсените решения*.

b	d	t	bdt_{10}	bdt_8	$C_{2\times2\times2}$	$Z_{2\times2\times2}$
0	0	0	0	0	1	1
0	0	1	1	1	1	3
0	1	0	10	2	1	2
0	1	1	11	3	0	0
1	0	0	100	4	1	5
1	0	1	101	5	0	0
1	1	0	110	6	1	4
1	1	1	111	7	1	6

b	r	br_{10}	br_6	$Z_{2\times3}$
0	0	0	0	3
0	1	1	1	1
0	2	2	2	2
1	0	10	3	6
1	1	11	4	4
1	2	12	5	5

4.1. Конструиране на критерии за двузначен избор (по Бул). Следният програмен фрагментът, разглеждан в контекста на горния диалог, постига решението на задачата $C_{2\times2\times2}$ за непротиворечивост на отговорите b, d и t, под формата на обобщения двузначен признак $C2 \times 2 \times 2$, тълкуван в булев смисъл.

```
int C2 × 2 × 2;
    C×2 × 2 × 2 = ! (!b && d && t || b && !d && t)
```

4.2. Конструиране на критерии за многозначен операторен избор. Следният фрагмент решава задачата $Z_{2\times2\times2}$, като конструира десетичен код за многозначен избор, запазващ комбинаторното разнообразие на входните данни:

```
int bdt10;
bdt10 = 100*b + 10* d + t;
switch (bdt10) { case   0: Z = 1; break;
                 case   1: Z = 3; break;
                 case  10: Z = 2; break;
                 case 100: Z = 5; break;
                 case 110: Z = 4; break;
                 case 111: Z = 6; break;
                 default : Z = 0; }
```

*По-богат анализ на същата етюдна задача може да бъде намерен в [2].

По аналогичен начин, на базата на десетичния код br10 = 10*b + r, е възможно да бъде намерено решението и на задачата $Z_{2\times3}$. Таблицата показва, че за разлика от задачата $Z_{2\times2\times2}$, в този вариант не съществува опасност от несъвместими отговори.

4.3. Конструиране на критерии за многозначен индексен избор. Следният фрагмент решава задачата $Z_{2\times2\times2}$, като конструира осмичен код за многозначен индексен избор, позволяващ справка в еднопараметричен справочник.

```
int bdt8, Z2 × 2 × 2 [8] = {1, 3, 2, 0, 5, 0, 4, 6 };
      bdt8 = 4*b + 2*d + t;
      Z = Z2 × 2 × 2 [bdt8]
```

По аналогичен начин, на базата на шестичния код br6, е възможно да бъде намерено решение и на задачата $Z_{2\times3}$:

```
int br6 , Z2 × 3 [6] = {3, 1, 2, 6, 4, 5};
      br6 = 3*b + t;
      Z = Z2 × 3 [br6]
```

4.4. Решение на задачата чрез многопараметричен индексен избор. Следният фрагмент решава задачата $Z_{2\times2\times2}$ чрез справка в трипараметричен справочник, осъществена с помощта на оригиналните стойности на трите отговора, използвани този път директно за един трипараметричен индексен избор:

```
int Z2 × 2 × 2 [2][2][2] = { { {1, 3}, {2, 0} },
                             { {5, 0}, {4, 6} } };
          Z = Z2 × 2 × 2 [b] [d] [t]
```

По същия високоефективен начин може да бъде решена и задачата $Z_{2\times3}$:

```
int Z2 × 3 [2] [3] = {{3, 1, 2}, {6, 4, 5}};
         Z = Z2 × 3 [b] [t]
```

Подходи, основани на подобни таблици, или пък използващи такива, крият риска от грешки, допуснати при съставянето на самите таблици. Интересно е, че същите тези подходи дават изключително удобно технологично решение и на този проблем. За целта в началния вариант на програмата, използваните в нея таблични стойности могат да бъдат подменени с фиктивни. Веднъж станала работоспособна, програмата полуфабрикат може и трябва, чрез краен брой експерименти с реални данни, емпирично да бъде прецизирана по отношение на своите таблици. Така реално съчетаваме конструирането на програмата с безусловно необходимата нейна проверка.

4.5. Алгебрична перифраза на формулата на Бул. Използването на нехомогенни изрази прави възможен синтеза на алгебричен израз, който директно реализира решението на задачата $Z_{2\times2\times2}$. Подходът съществено се основава на факта, че формулата на Бул продуцира "истина" за сметка на точно един от своите нормални дизюнкти. Тук, в своя алгебричен аналог, тя продуцира търсения резултат отново за сметка пак точно на едно от своите събираеми.

```
Z =   (!b && !d && !t) * 1
    + (!b && !d &&  t) * 3
    + (!b &&  d && !t) * 2
    + ( b && !d && !t) * 5
    + ( b &&  d && !t) * 4
    + ( b &&  d &&  t) * 6;
```

4.6. Двузначна индексна селекция на потенциалните отговори. Твърде екзотичен и с не особено висока експлоатационна ефективност, този подход се основава на една уникална и изключително интересна семантична и в моделна интерпретация на булевите стойности.

```
int product [ 2 ] = { 0, 0 };
   product [!b && !d && !t] = 1;
   product [!b && !d &&  t] = 3;
   product [!b &&  d && !t] = 2;
   product [ b && !d && !t] = 5;
   product [ b &&  d && !t] = 4;
   product [ b &&  d &&  t] = 6;
   Z = product [1]
```

Осезаемо е усещането, че използваме работно пространство с две позиции:

`product [1]` – за "качествената продукция", и `product [0]` – за "технологичния брак".

Благодарности. Изследването е частично финансирано по Договор № 120/2005 от Фонд "Научни изследвания" при СУ "Св. Климент Охридски".

ЛИТЕРАТУРА

[1] Н. Уирт. Алгоритми + структури от данни = програми. София , Техника, 1980.
[2] Д. Добрев. Една задача – сто решения. Цикъл от лекции. МОН, *Математика и информатика*, 1998, бр. 1–6.

Димитър Д. Добрев
Факултет по математика и информатика
СУ " Св. Климент Охридски"
бул. Дж. Баучер, № 5
1164 София
e-mail: dobrev@fmi.uni-sofia.bg

METHODOLOGY OF STATEMENT CONTROL AND DATA CHOICE IN PROCEDURE-ORIENTED PROGRAMMING

Dimitar D. Dobrev

We discuss the general problem for statement control and data choice in the procedure-oriented programming.
We propose some methods and techniques, based on the C Programming Language.