§ 59. Технологія побудови алгоритмів

Вивчивши цей параграф, ми:

познайомимося з технологією побудови алгоритмів;

дізнаємося, у чому полягають переваги проектування алгоритму методом «зверху донизу»;

з'ясуємо, які корисні якості має алгоритм, сконструйований з окремих модулів;

довідаємося, як у програмах здійснюється зв'язок між окремими модулями.

====59.1. Технологічний підхід до побудови алгоритму==========

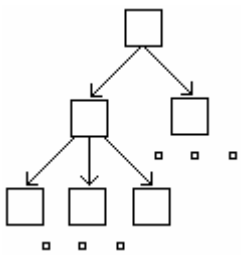
Будь-яке масове виробництво завжди спирається на технологію. Технологія дозволяє розподілити складний виробничий процес на окремі етапи й організувати роботу різних груп виконавців над кінцевим продуктом. Так створюються літаки й автомобілі, будуються споруди, виготовляються продукти харчування тощо.

З поширенням сфери застосування комп'ютера виникла індустрія програмних засобів, а з нею й технологія виробництва програмних продуктів.

Основою будь-якого програмного продукту є алгоритм. Ми познайомимося з ***технологією розробки алгоритму***, яка ґрунтується на трьох основних засадах:

- проектування алгоритму ***методом покрокової деталізації***, або ***методом «зверху донизу»***;
- організація алгоритму у вигляді відносно незалежних частин — модулів;
- застосування уніфікованих алгоритмічних структур.

Проектування алгоритму за методом покрокової деталізації розпочинається з визначення основних складових поставленої задачі. Назвемо їх підзадачами. Далі здійснюється аналіз кожної з виділених підзадач і визначаються їх складові і т. д. Урешті-решт задача буде подана як сукупність простих елементів, і в подальшій деталізації не буде потреби.



Головна перевага описаного методу полягає в тому, що в процесі проектування закладається правильність алгоритму: адже на кожному кроці фактично вирішується, чи можна цю задачу подати як сукупність більш простих задач. Поступове і цілком прозоре зменшення складності задачі зводить її до такого стану, коли алгоритм можна написати без помилок.

Спроектований у такий спосіб алгоритм складається з окремих модулів, які відповідають виділеним підзадачам.

Модуль — це логічна частина алгоритму, яка є відносно незалежною, має певне цільове призначення й вирішує тільки одну чітко сформульовану задачу.

Іноді в процесі проектування алгоритму з'ясовується, що одну й ту саму підзадачу необхідно вирішувати на різних етапах розв'язання задачі. Модуль для такої підзадачі створюється один раз, а використовується кожного разу, коли це потрібно.

Незалежність модулів дозволяє розподілити роботу зі складання алгоритму між різними групами виконавців, а в подальшому використовувати розроблені модулі як готові цег-линки для конструювання нових алгоритмів.

Технологічно доцільно при написанні алгоритму спиратися на застосування уніфікованих алгоритмічних структур — слідування, розгалуження, повторення. Це дозволяє використовувати стандартні прийоми програмування, що сприяє зменшенню помилок у програмах.

====59.2. Головний і допоміжні модулі. Формальні і фактичні параметри==========

Модулі, з яких складається алгоритм, розрізняються за рівнем. Модуль найвищого рівня ***(головний)*** охоплює всю поставлену задачу, модулі нижчих рівнів ***(допоміжні)*** відповідають її підзадачам.

Зв'язок між головним і допоміжними модулями встановлюється через вказівку про виконання допоміжного модуля. Ця вказівка називається *викликом модуля*.

Виклик повинен містити:

- ім'я модуля, що викликається;
- вхідні дані, які він має опрацювати;
- імена змінних, в які потрібно передати результати роботи модуля.

Кожний модуль є алгоритмом розв'язання певної задачі. Змінні, які є аргументами і результатами алгоритму, називають *параметрами модуля*. Модуль може викликатися для опрацювання різних наборів вхідних даних (аргументів) і передавати результати своєї роботи в різні змінні. Параметри модуля в його описі називають ***формальними***. Параметри, з якими відбувається виконання модуля, називають ***фактичними***. Отже, виклик модуля містить його ім'я й задає фактичні параметри для виконання модуля.

Фактичні параметри-аргументи можна задавати як безпосередньо значеннями, так й іменами змінних або виразами. Фактичні параметри-результати можуть бути тільки іменами змінних.

====59.3. Програма і процедури==========

При поданні алгоритму мовою програмування опис усіх модулів наводиться в програмі. Головний модуль складає тіло програми. Описи допоміжних модулів наводяться в описовій частині програми — після оголошення змінних головної програми.

Допоміжні модулі в мові Паскаль називаються ***процедурами*** (від англ. procedure — метод дій). Опис процедури цілком аналогічний опису програми, проте має суттєві відмінності:

- опис процедури розпочинають службовим словом **`procedure`** (а не **`program`**);
- після імені процедури в круглих дужках розміщують перелік формальних параметрів з оголошенням їх типів. Переліку імен змінних-результатів передує службове слово **`var`**;
- опис процедури завершується службовим словом **`end`** і крапкою з комою, а не крапкою.

Різницю між поданням програми і процедури покажемо на прикладі оформлення програми обчислення довжини відрізка (див. п. 58.3) у вигляді процедури.

```
procedure distance(xp, yp, xk, yk: real; var d: real);
    begin
        d := sqrt(sqr(xk - xp) + sqr(yk - yp));
    end;
```

Як бачимо, у процедурі відсутні оператори, пов'язані із введенням вхідних даних і виведенням вихідних даних.

Виклик процедури складається з імені процедури, що викликається, і переліку фактичних параметрів, який розміщується в дужках після імені процедури. Порядок і тип фактичних параметрів у виклику процедури має відповідати порядку й типу формальних параметрів в описі процедури.

Якщо опис процедури `distance` наведено в описовій частині програми, то з будь-якого місця програми можна викликати цю процедуру в такий, наприклад, спосіб:

```
distance(0, 5, 0, 17, z);
```

Унаслідок такого виклику змінна `z` одержить значення 12.

Процедурі можна передати фактичні параметри-аргументи не тільки у вигляді конкретних значень, а й через імена змінних або вирази.

Наприклад, при виконанні процедури `distance` за викликом

```
distance(xa, ya, xb, yb, ab);
```

поточні значення змінних `xa`, `ya` будуть використані як координати початкової точки відрізка, значення змінних `xb`, `yb` — як координати його кінцевої точки. Змінна `ab` одержить значення довжини відрізка. Якщо на момент виклику процедури `xa = 0, ya = 5, xb = 0, yb = 17`, то змінна `ab` набуде значення 12.

Важливо пам'ятати, що в програмі, яка викликає процедуру, тільки змінні, помічені у виклику як **`var`**, набувають нових значень. Усі інші змінні зберігають свої значення навіть у випадку, коли імена змінних в основній програмі і в процедурі збігаються.

====59.4. Задача Дідони (трикутний варіант) ==========

Продемонструємо застосування процедур на прикладі задачі Дідони.

За стародавньою легендою, Дідону, дочку і спадкоємицю могутнього царя, брати вирішили позбавити престолу. Вони вивезли Дідону на пустий берег моря, дали їй шкіру бика і сказали: «Побудуй собі місто на цій шкірі й царюй у ньому». Не розгубилася Дідона. Порізала вона шкіру на тонкі смужки, зв'язала їх у довгу мотузку й охопила цією мотузкою великий шматок прибережної землі. Здивувалися брати мудрості Дідони і не стали перешкоджати будівництву спочатку нової фортеці, а згодом і нового міста. Так був заснований Карфаген.

Сформулюємо задачу Дідони у спрощеному варіанті. Будемо вважати, що для закріплення шкіряної мотузки Дідона мала три кілочки. Припустимо також, що шкіра бика має площу 4 кв. метри, а ширина смужок, на які Дідона її розрізала, складає 1 мм. Отже, довжина шкіряної мотузки складає 4000 м. Задача полягає в тому, щоб визначити, у який спосіб і яку найбільшу площу може огородити Дідона за допомогою трьох кілочків і мотузки довжиною d = 4000 м.

Проаналізуємо постановку задачі. Ділянка землі, яку могла огородити Дідона, має форму трикутника. Одну його сторону (позначимо її AC) утворює лінія моря, дві інші — мотузка, натягнута на кілочки, встановлені в точках A, B і C. Задача зводиться до визначення трикутника найбільшої площі, у якого сума довжин двох сторін є заданою: AB + BC = d. Визначення трикутника в нашому випадку зводиться до визначення довжин сторін AB і BC.

Розробимо інформаційну модель задачі. Позначимо через `ac` і `ab`, відповідно, значення довжин сторін AC і AB. Очевидно, що значення `ac` і `ab` мають бути меншими за `d`. Довжину сторони BC позначимо через `bc`. Значення `bc` будемо знаходити за формулою: `bc = d - ab`. Площу трикутника (позначимо її через `s`) будемо визначати за трьома відомими довжинами його сторін за формулою Герона.

Алгоритм обчислення значення `s` складається з таких дій: задати значення `ac` і `ab`, знайти значення `bc`, обчислити `s`.

Розробимо комп'ютерну модель задачі. Змінні `ab, ac, bc, s` оголосимо як змінні дійсного типу, сталу величину `d` подамо як константу. Для обчислення площі трикутника скористаємося процедурою `Geron`.

```
program Didona;
 uses Crt;
 const d = 4000;
 var ab, ac, bc, s: real;
 procedure Geron(a, b, c: real; var s: real); {Подаємо опис процедури}
  var p: real;
  begin
  p := (a + b + c)/2;
  s := sqrt(p*(p - a)*(p - b)*(p - c));
 end;
```

```
begin
 clrscr;
 writeln('Уведіть відстань AC (< 4000): ');        {Уводимо вхідні дані:}
 readln(ac);                                       {вибрані значення довжин}
 writeln('Уведіть відстань AB (< 4000): ');        {двох сторін трикутника}
 readln(ab);
 bc := d - ab;                    {Знаходимо значення довжини третьої сторони}
 Geron(ab, bc, ac, s);                          {Обчислюємо площу трикутника}
 writeln('Площа: ', s:10:1, 'кв.м');                {Виводимо результат}
 readln;
end.
```

Як бачимо, структура тіла програми є цілком прозорою. Подробиці обчислення площі трикутника розкриваються в процедурі `Geron`.

Сплануйте проведення комп'ютерного експерименту для знаходження розв'язку задачі і встановіть значення.

ВИСНОВКИ

Основу технології розробки алгоритму складають: проектування алгоритму методом покрокової деталізації, організація його у вигляді відносно незалежних частин — модулів, застосування уніфікованих алгоритмічних структур. Метод покрокової деталізації дозволяє звести складну задачу до сукупності простих. Модульна організація алгоритму створює умови для розподілу роботи між окремими виконавцями і для використання раніше розроблених алгоритмів як готових блоків при побудові нового алгоритму. Застосування уніфікованих структур дозволяє стандартизувати прийоми його перетворення на програму. Усі три складові технології сприяють мінімізації помилок і одержанню правильного результату. В поданні алгоритмів мовою Паскаль роль модулів відіграють процедури. Кожна процедура має своє ім'я і список формальних параметрів. Для виконання процедури використовується її виклик, де формальні параметри заміщуються фактичними.

Контрольні питання та вправи

1. Що означає проектування алгоритму методом «зверху донизу»?

   а) поступове роздроблення задачі на все більш прості складові;
   б) запис алгоритму, починаючи з верхнього рядка і до самого нижнього;
   в) управління процесом проектування алгоритму з верхньої до нижньої ланки.

2. Модульна організація алгоритму — це:

   а) організація алгоритму з крупних блоків;
   б) організація управління алгоритмом за допомогою модулів;
   в) складання алгоритму з окремих відносно незалежних частин.

3. Виклик модуля містить:

   а) ім'я модуля, який викликає допоміжний;
   б) ім'я модуля, який викликається;
   в) вхідні дані, потрібні для виконання модуля;
   г) імена змінних для зберігання результатів виконання модуля.

4. У наведених нижче твердженнях пропущені слова *формальні* й *фактичні*. Вставте потрібні слова замість трикрапок.

а) … параметри присутні в описі процедури, а … — у її виклику;
б) … параметри використовуються в тілі процедури для опису дій, а … — для їх виконання;
в) через … параметри процедурі передаються значення аргументів і від неї отримуються значення результатів;
г) … параметри заміщують … в момент виклику процедури.

5. Процедура `summa(a, b: integer; var s: integer)` знаходить значення `s` як суму чисел `a` і `b`. Який оператор виклику цієї процедури є правильним і таким, що дозволить надати цілій змінній *x* значення суми чисел 10 і 5?

а) `summa(10, 5: integer; var x: integer);`
б) `summa(10, 5, x := s);`
в) `summa(10, 5, x);`
г) `summa(10, 5; x);`
д) `summa(5.0, 10.0, x);`
е) `summa(10, 5, var x).`

6. Процедура `average(n, m, k: integer; var x: real)` знаходить значення `x` як середнє арифметичне чисел `n`, `m` і `k`. Які оператори виклику цієї процедури не містять помилок, якщо всі змінні є змінними дійсного типу?

а) `average(4, 5, 6, z3);`
б) `average(10, 5.0, 20, s);`
в) `average(10, 5, –200, x);`
г) `average(426848, 15254, 16666, z);`
д) `average(25, 31, 23: integer; var z: real);`
е) `average(10+5, 89–45, 8*7, x);`
є) `average(75, 81/9, 3, x);`
ж) `average(sqr(2), sqr(3), sqr(4), seredne).`

7. Складіть процедуру, яка виводить на екран рядок із 20 зірочок, починаючи із заданої позиції. Скористайтеся цією процедурою для складання програми виведення на екран 5 різнокольорових рядків зірочок.

8. Складіть процедуру, яка виводить на задане місце екрана прямокутну рамку із зірочок. Скористайтеся цією процедурою для складання програми виведення на екран 5 різнокольорових рамочок.

9. Складіть процедуру виведення в задану позицію екрана зірочки заданого кольору. Скористайтеся цією процедурою для складання програми виведення на екран фігури з різнокольорових зірочок.

10. За допомогою програми `Didona` проведіть експеримент і визначте, яку площу могла огородити Дідона за допомогою трьох кілочків. Модифікуйте програму `Didona` для знаходження площі, яку можна огородити за допомогою чотирьох кілочків. Поясніть одержані результати. Дайте відповіді на запитання:

1) Якої форми набуває ділянка Дідони із збільшенням кількості кілочків?

2) Якщо мотузку можна було просто розкласти на землі, то як діяла мудра Дідона і на якій площі був заснований давній Карфаген?

11. Складіть програму визначення площі опуклого п’ятикутника, заданого координатами його вершин на площині. Скористайтеся процедурами `distance` (див. п. 59.3) і `Geron` (див. п. 59.4).

12. Складіть програму розв’язання задачі про ворону. На вершині дерева із золотими яблучками сидить ворона. Вона хоче вкрасти яблучко — одне з тих, що розсипані навколо дерева. Ділянку з яблуками огородили парканом. Яблука охороняє сторож. Яке яблуко слід схопити вороні, щоб якнайшвидше вилетіти за паркан? Висота дерева `h`, паркану — `z`, відстань від дерева до паркану — `r`.

Розгляньте два варіанти задачі:

1) швидкість польоту ворони вниз (з дерева до яблука) і вгору (з яблуком через паркан) є однаковою;

2) вгору ворона летить повільніше, ніж вниз.

Для складання програми скористайтеся процедурою `distance` (п. 59.3).

виклик допоміжного алгоритму, виклик процедури, головний алгоритм, допоміжний алгоритм, метод “зверху донизу”, метод покрокової деталізації, метод послідовного уточнення, модуль, процедура, структурний підхід, технологія побудови алгоритму, фактичний параметр, формальний параметр