Статијата прв пат е објавена во списанието Тангента на ДМ на Србија во 1995/96 година

СУМЕ И ПРОИЗВОДИ

Мирјана Ивановић, Бура Паунић
Природно-математички факултет, Нови Сад

У програмирању се често јавља потреба за програмима који израчунавају нумеричке вредности елемената низова, датих аналитичким изразом. Једна од класа низова погодних за израчунавање на рачунару су рекурентни низови, тј. низови облика $\{a_n\}$, где је n-ти елемент низа a_n дефинисан помоћу претходних елемената низа.

Најједноставнији случај рекурентних низова је када се вредност једног елемента низа може приказати као функција од вредности једног јединог претходног елемента што се може записати на следећи начин:

$$a_n = \begin{cases} a, & n = 0, \\ f(a_{n-1}), & n \geq 0, \end{cases}$$

где у функцији f осим елемента a_{n-1} могу да фигуришу и неке друге вредности које зависе од n. При томе се претпоставља да је број чланова низа

који треба да се израчуна коначан. Објаснимо поступак на једном једноставном примеру.

Пример 1. *Написати функцију која израчунава елементе низа* $\{x^n\}$.

За задати природан број n важи $a_n = x^n = x^{n-1} \cdot x = a_{n-1}x$ тј. наредни елемент низа се добија тако што се претходни елемент помножи са вредношћу x. Претпоставимо да је $0 \leq n \leq granica$, да је $0 \leq k \leq n$, при чему је узето да је $granica = 100$ и да је x цео број. Да би се формално одредио поступак за израчунавање вредности елемента низа најпре треба одредити функцијску зависност f између произвољна два суседна елемената низа тј. a_k и a_{k-1}. Веза се у овом случају најједноставније налази из односа a_k/a_{k-1} (за $x \neq 0$):

$$\frac{a_k}{a_{k-1}} = \frac{x^k}{x^{k-1}} = x \qquad \text{па је} \qquad a_k = \begin{cases} a_{k-1}x, & k > 0, \\ 1, & k = 0. \end{cases}$$

У наставку је дата једноставна функција за израчунавање вредности наредног елемента датог низа у *Pascal* језику. Променљивом ok омогућава се контрола прекорачења опсега целих бројева при израчунавању вредности наредног члана низа.

```
function an(n: integer;
            var ok: boolean): integer;
var
  i, rezultat : integer;
begin
   ok := true;
   if (x = 0) and (0 <= n) then begin
     ok := false;
     an := 0
   end
   else begin
     if x = 0 then
       an := 0
      else
        if (n = 0) or (x = 1) then
          an := 1
        else
          if n = 1 then
            an := a
          else begin
            rezultat := 1;
            for i := 1 to n do
              rezultat := rezultat * x
              an := rezultat;
```

```
        end
    end
end; (*  an  *)
```

Када је поступак за израчунавање наредног елемента низа једноставан (тј. функција f) нема потребе за увођењем посебне функције за његово израчунавање као што је то код овог низа случај. У супротном се за поступак израчунавања може увести посебна функција.

Суме и производи

Важан специјалан случај рекурентних релација јесу суме и производи. Нека је са a_k означен општи елемент низа тј. поступак за израчунавање вредности k-тог елемента низа. Са S_n ћемо означавати суму од n елемената низа при чему је низ задат изразом a_n. Писаћемо

$$S_n = a_1+a_2+\cdots+a_n = \sum_{k=1}^{n} a_k, \quad \text{или рекурентно} \quad S_n = \begin{cases} 0, & n = 0, \\ S_{n-1} + a_n, & n > 0. \end{cases}$$

Слично се дефинише и производ елемената низа b_n

$$P_n = b_1 \cdot b_2 \cdot \ldots \cdot b_n = \prod_{k=1}^{n} b_k, \quad \text{или рекурентно} \quad P_n = \begin{cases} 1, & n = 0, \\ P_{n-1} \cdot b_n, & n > 0. \end{cases}$$

Већина програмских језика омогућује да се суме и производи израчунавају коришћењем петљи, најчешће варијацијом следећих општих поступака. Структура програма је следећа помоћу `while` петље:

```
suma := 0.0;
i := 1;
while i <= n do begin
  izracunati sabirak ai;
  suma := suma + ai;
  i := i + 1
end;
```

```
proizvod := 1.0;
i := 1;
while i <= n do begin
  izracunati cinilac bi;
  proizvod := proizvod * bi;
  i := i + 1
end;
```

или слична са `for` или `repeat` петљом.

Поступак израчунавања суме чланова низа илустроваћемо на следећим примерима:

- Сума квадрата
- Сума елемената облика x^{k^2}

- Сума факторијела

Пример 2. *Написати функцију која израчунава суму квадрата природних бројева.*

Треба написати функцију која израчунава $S_n = 1^2 + 2^2 + 3^2 + \cdots + n^2 = \sum_{k=1}^{n} k^2$, $(0 \leq n \leq 100)$.

Користићемо општи поступак за израчунавање суме са `while` петљом, при чему ћемо водити рачуна о прекорачењу опсега и одговарајућу контролу ћемо уврстити у услов `while` петље. Величину сабирка не морамо проверавати ($100^2 = 10000 < maxint$), па се добија функција:

```
function ZbirKvad(n: integer;
                  var ok: boolean): integer;
const
  donjaGr = 0;
  gornjaGr = 100;
var
  i, sabirak, suma: integer;
begin
  if (donjaGr <= n) and (n <= gornjaGr) then begin
    ok := true;
    suma := 0;
    i := 1;
    while ok and (i <= n) do begin
      sabirak := sqr(i);
      ok := maxint - suma > sabirak;
      if ok then
        suma := suma + sabirak;
      i := succ(i)
    end;
    ZbirKvad := suma
  end
  else begin
    ok := false;
    ZbirKvad := 0
  end
end; (*  ZbirKvad  *)
```

У овом примеру се може убрзати израчунавање следећег сабирка, јер је сабирање брже од множења. Како је $(k+1)^2 = k^2 + 2k + 1$ то се нови сабирак ($(k+1)^2$) може израчунати тако да се на стари сабирак (k^2) дода повећање ($2k+1$), а само повећање се такође у сваком кораку повећава за 2.

Пример 3. *Написати процедуру која израчунава суму чији су сабирци облика* x^{k^2}.

Треба написати процедуру која за дати реалан број x и цео број n израчунава суму елемената облика x^{k^2}, $1 \leq k \leq n$ тј.

$$S_n = x + x^4 + x^9 + \cdots + x^{n^2} = x^{1^2} + x^{2^2} + x^{3^2} + \cdots + x^{n^2} = \sum_{k=1}^{n} x^{k^2},$$

користећи множење. Како је степеновање специјалан случај производа, то за израчунавање вредности x^{k^2} можемо користити једну посебну петљу:

```
suma := 0.0;
for k := 1 to n do begin
  sabirak := 1.0;
  for i := 1 to sqr(k) do
    sabirak := sabirak * x;
  suma := suma + sabirak
end;
```

Овај поступак је једноставан, али захтева за сваки степен од k, k^2 множења, што је изузетно неефикасно.

Да бисмо израчунавања учинили ефикасним применићемо општи поступак за израчунавање вредности елемената низа, $a_k = f(a_{k-1})$. Најпре треба наћи функцијску везу f између два суседна елемента. За $x \neq 0$ и $a_k = x^{k^2}$ добијамо:

$$\frac{a_k}{a_{k-1}} = \frac{x^{k^2}}{x^{(k-1)^2}} = x^{2k-1}$$

па је

$$\begin{aligned} a_k &= x^{2k-1} a_{k-1}, \quad k > 0 \\ a_0 &= 1 \end{aligned}$$

Ако се сад и израчунавање x^{2k-1}, такође реализује општим поступком за израчунавања елемента низа, а не директним множењем добијамо:

$$x_k = x^{2k-1}, \quad \frac{x_k}{x_{k-1}} = \frac{x^{2k-1}}{x^{2k-3}} = x^2, \qquad \text{тј.} \qquad \begin{aligned} x_k &= x^2 x_{k-1}, \quad k > 1, \\ x_1 &= x. \end{aligned}$$

Коначно се добија да се x^{k^2} израчунава коришћењем низова: a_k и x_k. За израчунавање k-тог елемента низа сада је потребно само $2k$ множења уместо k^2 множења како је наведено на почетку. Дакле

$$a_k = \begin{cases} x_k a_{k-1}, & k > 0, \\ 1, & k = 0, \end{cases} \qquad x_k = \begin{cases} x^2 x_{k-1}, & k > 1, \\ x, & k = 1. \end{cases}$$

Израчунавање елемената ова два низа може се реализовати једним од следећих програмских фрагмената (где се због једноставности не води рачуна о прекорачењу опсега). Фрагмент са десне стране је нешто лошији због тога што неизбежно настаје грешка при дељењу код израчунавања вредности променљиве a.

```
if x = 0.0 then
  suma := 0.0
else begin
  xk := x;
  a := 1.0;
  x2 := sqr(x);
  for k := 1 to n do begin
    suma := suma + a
    xk:= xk * x2;
    a := a * xk
  end
end;
```

```
if x = 0.0 then
  suma := 0.0
else begin
  xk := x;
  a := 1.0 / x;
  x2 := x*x;
  for i := 1 to n do begin
    a := a * xk;
    xk := xk * x2;
    suma := suma + a
  end
end;
```

У случају када је изложилац k^2 добили смо поступак за степеновање броја који је ефикаснији од директног множења, па је то индикација да се потражи и ефикаснији поступак за израчунавање x^k што се оставља читаоцима као задатак за вежбу.

Производ свих узастопних природних бројева од 1 до n се назива **факторијел броја** n и то се записује на следећи начин: $n! = n \cdot (n-1) \cdot (n-2) \ldots 2 \cdot 1$ или рекурентно $n! = n \cdot (n-1)!$ и $1! = 1$. Уобичајено је да се дефинише да је $0! = 1$.

Пример 4. *Написати процедуру која израчунава вредност следеће функције:*

$$S_n = \sum_{k=1}^{n} (-1)^k (2k+1)! y^{2k-1}$$

где је $-2 < y < 2$, а $1 \leq n \leq 100$.

Применимо општи поступак за израчунавање сума ($S_n = S_{n-1} + a_n$, $S_1 = a_1$), када је сабирак $a_k = (-1)^k (2k+1)! y^{2k-1}$.

Веза a_k и a_{k-1} се добија из количника $\frac{a_k}{a_{k-1}}$ тј. после сређивања количника добијамо $a_k = -2k(2k+1)y^2 a_{k-1}$ и $a_1 = -6y$.

Ако се искористи да је за $y = 0.0$, $f(0.0, n) = 0.0$, добија се следећа функција за израчунавање вредности полазне суме:

```
function f(y: real;
           n: integer;
           var ok: boolean): real;
const
  ygr = 2.0;
  ndgr = 1;
  nggr = 100;
var
  k : integer;
  sabirak, zbir, y2 : real;
begin
```

```
  ok := true;
  if (ndgr <= n) and (n <= nggr) then
    if y = 0.0 then
      f := 0.0
    else begin
      sabirak := - 6.0 * y;
      zbir := sabirak;
      y2 := sqr(y);
      for k := 2 to n do begin
        sabirak := - sabirak*2*k*(2*k + 1) * y2;
        zbir := zbir + sabirak
      end;
      f := zbir
    end
  else
   ok := false
   f := 0.0
end; (*  f  *)
```

ЗАДАЧИ

1. Одредити поступак и написати програм за израчунавање x^n, за цео број n.

2. *Одредити ефикасан поступак за израчунавање вредности* x^n *множењем* (n природан број). (Упутство: Користити

$$x^n = \begin{cases} x, & n = 1, \\ (x^k)^2, & n = 2k, \\ x(x^k)^2, & n = 2k + 1. \end{cases}$$

3. Написати програм за израчунавање збира квадрата и кубова природних бројева ($n < 100$) у којима се не користи множење.

4. Израчунати вредност полинома

$$S_n(x) = \sum_{k=1}^{n} \frac{(-1)^k (2k+1)! x^{2k-1}}{(k!)^2}$$

где је $-2 < x < 2$ и $1 \leq n \leq 100$.