# СОНИ АНИҚ БЎЛМАГАН, ИЧМА-ИЧ ЖОЙЛАШГАН ЦИКЛЛИ СТРУКТУРАЛАРНИ ДАСТУРЛАШ МУАММОЛАРИ

**О.А.МАМАТҚУЛОВ**

*В статье рассматривается вопрос о создании программмных структур n вложенных уровней циклов с использованием рекурсии.*

*In the column described the way of constructing n-level inner loops using recursion in software development.*

Кўпчилик бошловчи дастурчиларда сони олдиндан аниқ бўлмаган, ўзаро ичма-ич жойлашган цикллардан иборат структураларни тузиш қийинчилик туғдиради. Аксига олиб, кўплаб дастурлаш масалалари шу ҳолатга келиб қолади. Бу муаммони ҳал қилишнинг дастурчи учун осон бўлган усулларидан бири рекурсиядан фойдаланишдир.

Маълумки, бевосита ёки билвосита ўзини ўзи чақириб, қайтадан ишга туширувчи қисм дастур (функция, процедура, метод) рекурсив дейилади [1,2,3]. Масалан:

```
void myRecFunc(int n) {

    if (n == n0)
        return;
    . . .
    myRecFunc(m);
    . . .
}
```

Бунинг ўзига хослиги ҳар бир янги чақирув мутлақо янги функцияни ишга туширади. Дастур матнида ёзилган функция битта бўлса ҳам, ҳар бир чақирилганда хотирадан янги локал ўзгарувчилар (ва параметрлар) учун жой ажратилади. Рекурсив функцияда базавий холат (if (n == n0)), албатта, ҳисобга олиниши шарт, акс ҳолда, функция ўзини қайта ишга туширишдан тўхтамайди.

Адабиётларда рекурсия факториални ҳисоблаш, фиббоначи сонлари кетма-кетлигини ҳисоблаш каби мисоллар билан тушунтирилади [1,2,3].

Рекурсия ёрдамида ёзилган дастур қисмини бошқа усулларда ҳам муқобил дастурлаш мумкин. Масалан, стек структурасидан фойдаланиб локал ўзгарувчиларни сақлаб бориш мумкин. Лекин бу ёки бошқа усулда ёзилган дастурга нисбатан рекурсиядан фойдаланиб ёзилган дастур дастурчи учун кўпроқ тушунарли кўриниш олади. Бунга асосий сабаб шуки, воқеликда учрайдиган кўплаб улкан муаммолар кичикроқ, лекин ечилиш усули бир хил ёндашишни талаб қиладиган муаммочаларни ечиш орқали ҳал қилиниши мумкин ва айни шу нарса инсонга тушунарлиликни таъминлайди.

Лекин баъзи ҳолларда рекурсив функциялар динамик дастурлаш ёки бошқа итерацион усуллардаги ечимларга нисбатан самарасиз, секин ишлайдиган бўлиши мумкин. Бу, кўпинча, бир хил параметрлар учун функциянинг қиймати бир марта эмас, кўп марта қайтадан ҳисобланадиган ҳолатларда юзага келади. Бундай ҳолатларда динамик дастурлаш усули ёрдамга келади [1,2,3].

Аниқ мисолда кўриб чиқайлик. Масалан, 1,2,3 ва 4 сонларидан тузилган барча тўпламларни кетма-кет санаб ўтувчи дастур тузиш талаб этилган бўлсин (1234, 1243, . . . 4321). Кўплаб дастурларни тузишда, масалан, барча комбинацияларни кўриб, улар ичидан оптималини топиш билан боғлиқ муаммоларнинг ҳаммасида, кетма-кет барча вариантларни кўриб чиқадиган дастур шакллантириш талаб этилади.

Дастурда керак бўладиган ёрдамчи функцияларни келтирайлик.

```
#include <cstdlib>
#include <iostream>
using namespace std;

bool inArray(int *a, int n, int x)
   {for(int i=0; i<n; i++) {
       if (a[i] == x)
          return true;
}
return false;
}

void prn(int *a, int n) {
for(int i=0; i<n; i++) {
cout << a[i];
}
cout << endl;
}


. . .
```

Функцияларнинг биринчиси – inArray() функцияси x сонининг a массивнинг дастлабки n та элементлари орасида борлигини аниқлаб беради ва ундан бир марта ишлатилган сонни иккинчи марта қайта ишлатмаслик учун текширишда фойдаланамиз.

Иккинчи функция – prn() дан эса берилган массивнинг элементларини чоп этиш учун фойдаланамиз.

Энди бевосита қўйилган масалани ҳал қилувчи дастурни кўришга ўтайлик. Энг биринчи хаёлга келадиган дастур структураси қуйидагича кўриниш олади (C++ дастурлаш тили):

```
. . .

int main() {
   int a[4];

   for(a[0]=1; a[0]<=4; a[0]++)                  // 1
      for(a[1]=1; a[1]<=4; a[1]++)               // 2
         if (!inArray(a, 1, a[1]))
            for(a[2]=1; a[2]<=4; a[2]++)         // 3
               if (!inArray(a, 2, a[2]))
                  for(a[3]=1; a[3]<=4; a[3]++)  // 4
                     if (!inArray(a, 3, a[3]))
                        prn(a, 4);

   return EXIT_SUCCESS;
}
```

Бу ечимнинг асосий камчилиги шуки, унда ичма-ич жойлашган цикллар сони ўзгарувчан бўла олмайди, олдиндан (дастурлаш вақтидан) сони аниқ бўлиши талаб қилинади. Чунки дастур матнида ичма-ич жойлашган циклли конструкциялар жойлашган.

Энди масалани сони олдиндан аниқ бўлмаган ҳолат учун кўрайлик. Айтайлик, 1 дан n гача бўлган сонлардан тузилган барча тўпламларни кетма-кет санаб ўтувчи дастур тузиш талаб этилган бўлсин.

Тўплам элементларини a номли массивда сақлаб борамиз. Массив элементлари турли хил сонлар билан тўлгач, битта комбинацияга эга буламиз ва уни чоп этиш мумкин.

```
. . .

void myRecFunc(int *a, int n, int k) {
   if(k == n) {
      prn(a, n);
      return;
   }

   for(int i=1; i<=n; i++) {
      if (!inArray(a, k, i)) {
         a[k] = i;
         myRecFunc(a, n, k+1);
      }
   }
}


int main() {

   int n;
   cout << "n? ";
   cin >> n;

   int *a = new int[n];

   myRecFunc(a, n, 0);

   delete[] a;

   return EXIT_SUCCESS;
}
```

Бу қандай ташкил этилаётганлигини чуқурроқ қараб чиқайлик. Рекурсив функциямиз учта параметрга эга бўлиб, биринчиси – массив, иккинчиси – n, учинчиси рекурсив функция нечанчи марта чақирилаётган эканлигининг ҳисобини юритиш учун қўшимча параметр. Айнан мана шу учинчи параметр ҳозиргача ичма-ич жойлаштирилган цикллар сони бўлиб, рекурсиянинг базавий ҳолатини аниқлашда ёрдам беради.

Ушбу ечимда ичма-ич жойлашган цикларни ташкил қилиш учун рекусив функция myRecFunc() дан фойдалнилмоқда. Аслида, бу функция танасидан битта цикл ўрин эгаллаган. Лекин ўша цикл ичидан туриб, функция ўзини қайтадан ишга туширмоқда. Натижада цикллар ичма-ич жойлашганлик эффектини бераяпти. Буни қуйидагича тасвирлаш мумкин:

```
myRecFunc(a, n, 0) {
   for(int i=1; i<=n; i++) {              // 1
      a[0] = i
      myRecFunc(a, n, 1)
         for(int i=1; i<=n; i++) {        // 2
            a[1] = i
            myRecFunc(a, n, 2)
               for(int i=1; i<=n; i++) {  // 3

                  . . .                  // ...

                   myRecFunc(a, n, n)
                      prn(a)
                      return
```

Шундай қилиб, сони олдиндан аниқ бўлмаган, ўзаро ичма-ич жойлашган циклли структураларни рекурсия ёрдамида дастурлаш жуда осон экан. Дастурлашда рекурсиядан фойдаланиш дастурчига тушунарлиликни тақдим қилиши билан бошқа усуллардан афзал. Бундан ташқари, кўрилган ҳолат учун рекурсив функция самарадорликнинг камайишига ҳам олиб келмаяпти.

**Адабиётлар:**

1. Michael T. Goodrich, Roberto Tamassia, David M. Mount. Data Structures and Algorithms in C++. 2nd ed. – Hoboken, New Jersey: – John Wiley & Sons Inc., 2011. – 714 p.

2. Шень А. Программирование: теоремы и задачи. – 2-е изд., испр. и доп. – М.: МЦНМО, 2004. – 296 с.

3. Каррано Ф.М., Причард Дж.Дж. Абстракция данных и решение задач на C++. Стены и зеркала, 3-е изд.: Пер. с англ. – М.: Вильямс, 2003. – 848 с.

*Андижон давлат университети,*
*2012 йил 4 июнда қабул қилинган*
*(Тақризчи: п.ф.н. А.Медатов)*