6. Алгоритми и програми за работа с подпрограми

Задача f_1: Да се изчисли стойността на израза:
$c = (a+b)^{(k+m)} + (a.b)^{m}$, където a, b са реални числа, различни от 0, а k, m – цели числа.
Да се реализира подпрограма, която да повдига произволно реално число x, различно от 0, в произволна цяла степен n (x^n).

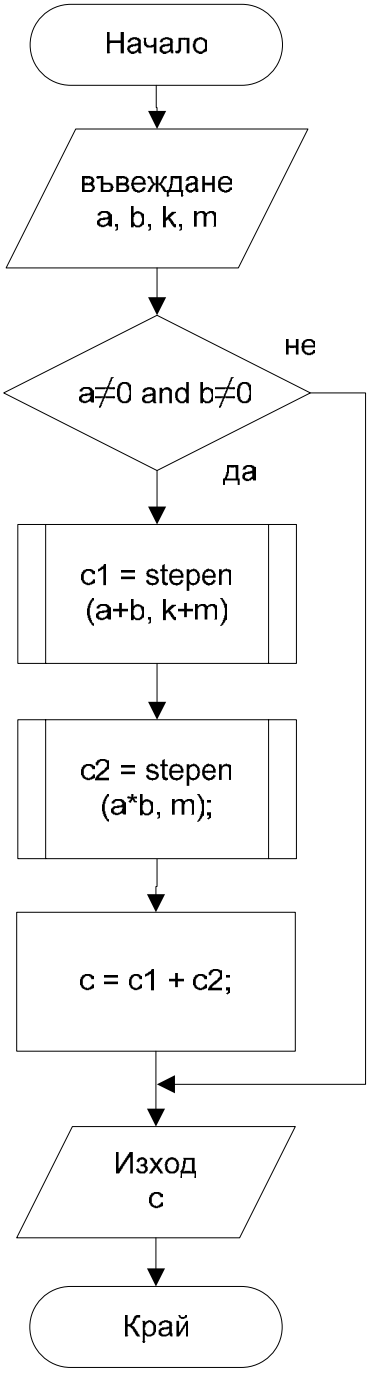
Решение:

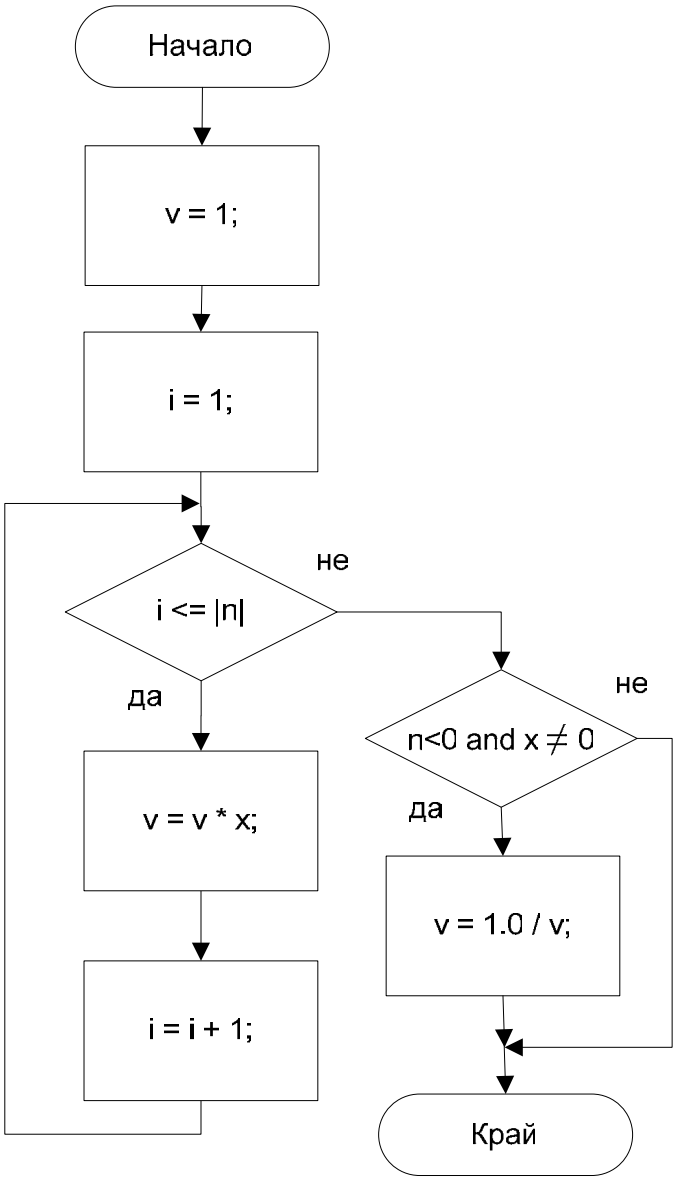
В главната програма се въвеждат входни данни за **a**, **b**, **k**, **m**. За пресмятане на израза се извиква два пъти функцията за степенуване. Извежда се крайния резултат.

Подпрограмата-функция е за повдигане на произволно реално число **x**, на произволна цяла степен **n**. На практика това означава да се умножи числото **x** само по себе си **n** пъти и функцията трябва да реализира пресмятане на произведение. За да се организира правилно цикъла за намиране на произведението, горната граница на брояча трябва да е положително число. Затова за горна граница се използва абсолютната стойност на степента **n**. Трябва да се съобрази също, че ако степента **n** е отрицателна, резултатът ще е равен на реципрочната стойност на това произведение. Необходимо е да се направи проверка и дали основата на степента е различна от 0.

Използват се следните променливи:
a и **b** – реални стойности;
k и **m** – цели стойности;
stepen – подпрограма-функция, която получава за вход реално число **x** и цяла степен **n**. За резултат пресмята x^n, т.е. стойността на повдигането на **x** на степен **n**;
x – формален параметър за основата на степента;
n – формален параметър за степенния показател;
i – управляваща променлива за цикъл;
v – променлива в подпрограмата, в която се получава резултата от умножението и степенуването.

Блок схемата на задачата трябва да съдържа и двата алгоритъма – на главната програма и на подпрограмата.





функция stepen(x,n)

1) **Програма на Pascal**

```
program f_1;
var
  a, b, c: real;
  k, m: integer;
{декларирane на подпрограма-функция за повдигане на
произволно число x в цяла степен n}
function  stepen(x:real; n:integer):real;
  var  i : integer;
       v: real;
  begin
    v:=1;    {начална стойност на произведението}
```

```
    for i:= 1 to abs(n) do
         v:=v*x;
    if (n<0) and (x<>0) then
        v:= 1.0/v;
    stepen := v
{ името на функцията присвоява изчислената стойност v}
  end;       {край на описанието на функцията }
begin        {главна програма}
  writeln('въведете стойност за a и b');
  readln (a, b);
  writeln('въведете стойност за k и m');
  readln (k, m);
  if (a<>0) and (b<>0) then
  begin
     c:= stepen(a+b, k+m) + stepen(a*b, m);
{ако a и b са различни от 0, функцията се извиква с
фактически параметри според условието на задачата}
     writeln('c=', c:9:3);
  end;
  readln;
end.
```

2) **Програма на C**

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
/*   дефиниция на функция за повдигане на произволно
число x в цяла степен n   */
double  stepen(double x, int n)
{   int   i;
    double v = 1;   // произведение и начална стойност
    for (i=1; i<=abs(n); i++)
         v *= x;
    if (n<0 && x!=0)
         v = 1.0/v;
    return v;
}           // край на описанието на функцията
// главна функция
int main(int argc, char *argv[])
{   float  a, b;
    double  c;
    int  k, m;
    printf("Въведете стойност за a и b = ");
    scanf("%f%f", &a, &b);
```

```
    printf("Въведете стойност за k и m = ");
    scanf("%d%d", &k, &m);
    if (a!=0 && b!=0)
    {   c = stepen(a+b, k+m) + stepen(a*b, m);
        printf("Резултат c = %e \n", c);
    }
    system("PAUSE");
    return 0;
}       // край на главната функция
```

3) **Програма на C++**

```
#include <cstdlib>
#include <iostream>
#include <cmath>

using namespace std;

// деклариране на функцията – прототип:
double  stepen(double x, int n);

// главна функция
int main(int argc, char *argv[])
{   double  a, b, c;
    int  k, m;
    cout << "Въведете стойност за a и b = ";
    cin >> a >> b;
    cout << "Въведете стойност за k и m = ";
    cin >> k >> m;
    if (a!=0 && b!=0)
    {   c = stepen(a+b, k+m) + stepen(a*b, m);
        cout << "Резултат c =" << c << endl;
    }
    system("PAUSE");
    return EXIT_SUCCESS;
}       // край на главната функция

/*   дефиниция на функция за повдигане на произволно
число x в цяла степен n   */
double  stepen(double x, int n)
{   int   i;
    double v = 1;    // произведение и начална стойност
    for (i=1; i<=abs(n); i++)
        v *= x;
    if (n<0 && x!=0)
        v = 1.0/v;
    return v;
}           // край на описанието на функцията
```

Задача f_2: Масив p[33] съдържа произволни реални стойности. Да се изчисли стойността на израза:
S = s1 + s2 +s3, където
s1 = p[1]+p[2]+ … +p[8];
s2 = p[5]+p[11]+ … +p[20];
s3 = p[25]+p[26]+ … +p[33].
За изчисляване на частичните суми да се реализира под-програма–функция.

Решение:

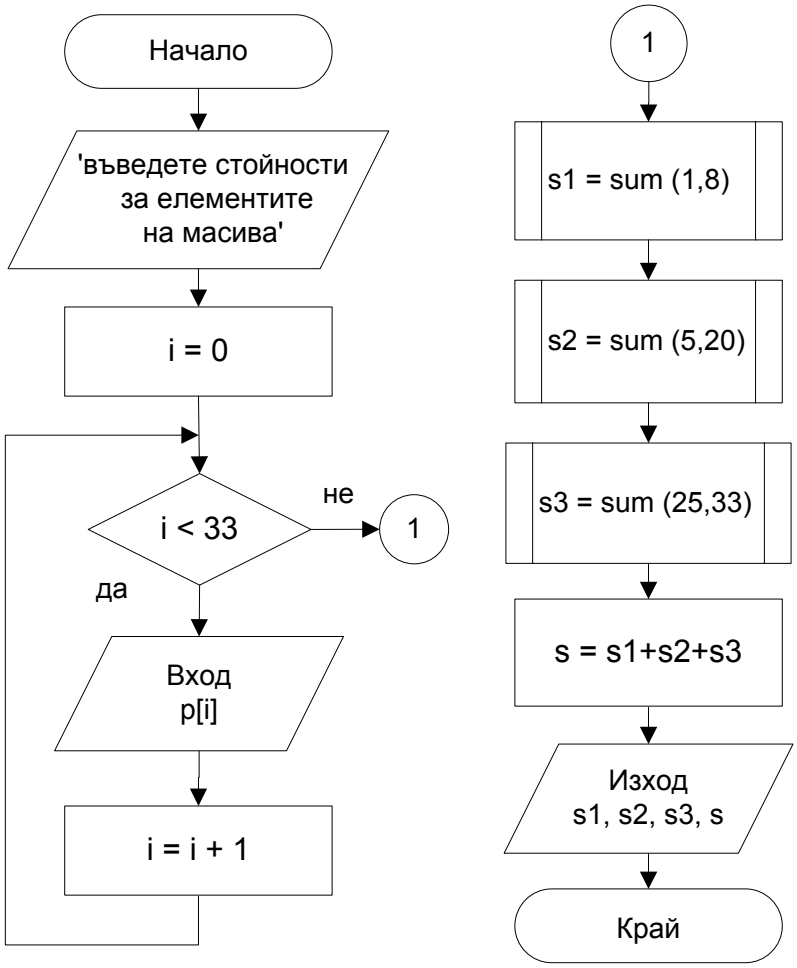
В задачата се изисква пресмятането на три суми на елементи от масив. Затова е удобно алгоритъмът за пресмятане на сума да се обособи в подпрограма-функция, която в програмата да се извиква три пъти за трите отделни суми. Частичните суми са за различни диапазони от масива, т.е. различен е началният и крайният индекс на елементите, които се сумират. Следователно при всяко извикване функцията трябва да получава информация за стойностите на началния и крайния индекс на елементите, които определят зададения диапазон. Това се постига чрез използване на подходящи формални параметри в описанието на функцията и необходимите им стойности за фактически параметри при използването (извикването).

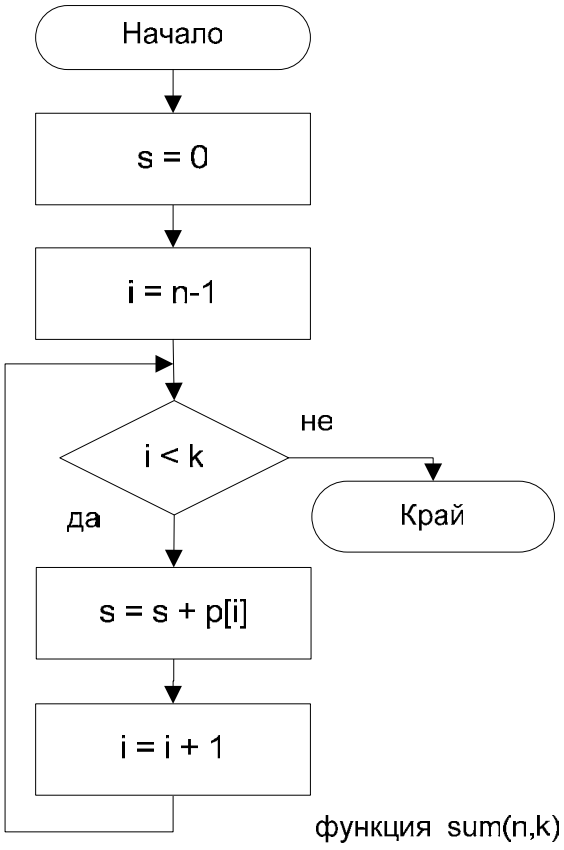
В примера масивът **p** е дефиниран като глобален в началото на програмата. Така той се използва в главната програма и във функцията.

Използват се следните величини:

p – масив;
i – текущ индекс на елемент на масив и управляваща променлива за цикъл;
s1, s2, s3 – частични суми за различните диапазони;
s – променлива за резултат сумата на трите частични суми;
sum –подпрограма-функция за намиране на сума в масив;
n – формален параметър на подпрограмата-функция за начален номер на елемент от диапазона за сумиране;
k - формален параметър на подпрограмата-функция за крайния номер на елемент от диапазона за сумиране.
s – във функцията променлива, в която се натрупва сумата на елементите от зададения диапазон. Резултат от функцията.

Блок-схемата на алгоритъма представя алгоритмите на главната програма и на функцията за сумиране.


Начало
'въведете стойности
за елементите
на масива'
i = 0
i < 33
не
1
да
Вход
p[i]
i = i + 1
1
s1 = sum (1,8)
s2 = sum (5,20)
s3 = sum (25,33)
s = s1+s2+s3
Изход
s1, s2, s3, s
Край


Начало
s = 0
i = n-1
i < k
не
Край
да
s = s + p[i]
i = i + 1
функция sum(n,k)

Програма на C++

```
#include <cstdlib>
#include <iostream>

using namespace std;

double  p[33];                  // глобални параметри
int i;
// предварително деклариране на функцията, прототип:
double sum (int n, int k);

int main(int argc, char *argv[])
{  double s1, s2, s3, s;
   cout << "Въведи 33 числа:\n";
   for (i=0; i<33; i++)
      cin >> p[i];
   s1 = sum(1,8);
   s2 = sum(5,20);
   s3 = sum(25,33);
   s = s1 + s2 + s3;
   cout << "Сума 1 = " << s1 << endl;
   cout << "Сума 2 = " << s2 << endl;
   cout << "Сума 3 = " << s3 << endl;
   cout << "Резултат S = " << s << endl;
   system("PAUSE");
   return EXIT_SUCCESS;
}
// дефиниция на функция за сума в диапазона n,k
double sum (int n, int k)
{  double  s=0.0;
   for (i=n-1; i<k; i++)
      s += p[i];
   return s;                    // връща резултат сумата
}
```

Идея: За да е по-универсална функцията **sum** тя трябва да може да се използва и за различни едномерни масиви (например два масива **p1** и **p2**). Тогава като параметър на функцията се подава и кой е обработвания масив. Промени:

- заглавието на функцията, имаща параметър едномерен масив **x**, може да има вида:

```
double sum(double *x, int n, int k)
```

- във функцията се обработва масив с името **x**
- извикването на функцията **sum** за различни масиви е:

```
s1 = sum(p1, 1, 8);
s2 = sum(p2, 5, 20);
```

Задача f_3: От масив mas[6,5] да се прехвърлят всички положителни елементи в едномерен масив. Обработката да е с подпрограма, връщаща като резултат едномерен масив и броя на елементите в него.

Решение: В тази задача, се обработва двумерен масив. Той може да е глобален параметър, както масивът в предишния пример, или за да може да се работи с различни масиви, масивът да се предава като формален параметър в подпрограмата. Подпрограмата трябва да има дефиниран формален параметър и за едномерния масив, в който се прехвърлят резултатите.

Дефинирането на формален параметър за име на едномерен масив в език C/C++ е като указател ***x** или чрез **x[]** като тук размерност може да не се посочва. И в двата варианта при извикването на подпрограмата фактически параметър е името на съответния едномерен масив. Така подпрограмата получава адреса на началото на масива и достъп до клетките в паметта, в които той е разположен. За двумерен масив като формален параметър, обаче, е необходимо да се посочи конкретно дефинирания размер на масива, за да се знае как се разполагат елементите в паметта, т.е. по колко елементи на ред.

В подпрограмата се обработва двумерен масив и се прехвърлят в едномерен масив положителните стойности. Броят на прехвърлените в масива числа също е резултат от подпрограмата-функция, връщан чрез оператор return.

Използват се следните величини:

M, N – глобални константи за размери на двумерния масив;

mas – двумерен масив;

p – едномерен масив с резултати – положителните елементи;

i и **j** – текущи индекси за ред и стълб, както и управляващи променливи за цикъл;

br – брой на положителните елементи в резултантния масив;

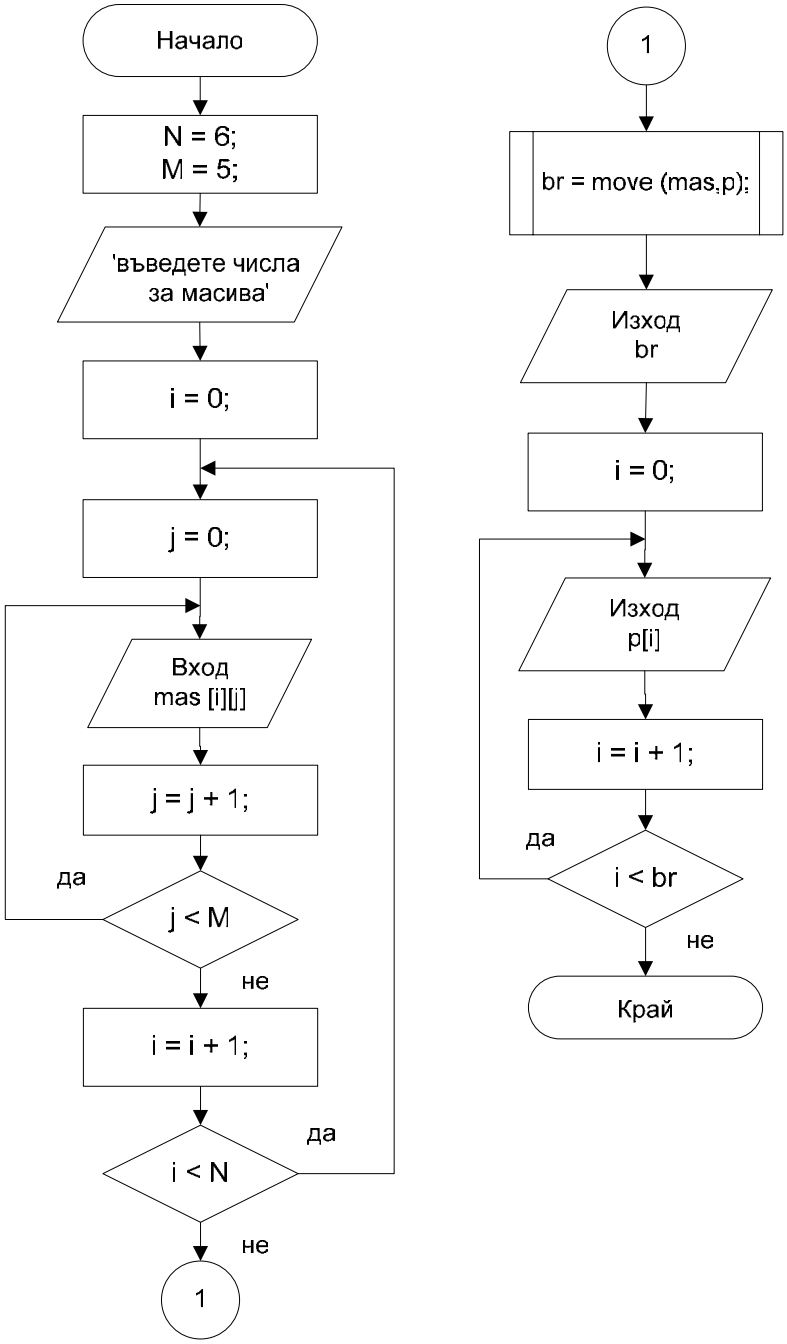
move – име на подпрограма-функция за прехвърляне на положителните елементи от двумерен масив в едномерен;

d – формален параметър в подпрограмата за обработван двумерен масив;

x - формален параметър за резултантен едномерен масив;

k - в подпрограмата брояч за индекса в едномерния масив;

Блок-схемата на алгоритъма е представена в обобщен вид. Представена е само главната програма, в която се посочва мястото на извикването на подпрограмата. Обработката на данни, която извършва подпрограмата, е позната и е разгледана в предишни примери.



Програма на C++

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

const int N=6, M=5;  // константи за брой ред и стълб
```

```
// прототип на функция за прехвърляне на числа
int move(double d[N][M], double x[]);

int main(int argc, char *argv[])
{   double mas[N][M], p[N*M];
    int i, j, br;
    cout << "Въведи " << N*M << " числа: \n";
    for (i=0; i<N; i++)
      for (j=0; j<M; j++)
        cin >> mas[i][j];
    br = move(mas, p);        // извикване на функция
    cout << "Брой >0   br = " << br << endl;
    cout << "\n Нов масив p[" << br << "]: \n";
    for (i=0; i<br; i++)
       cout << p[i] << "  ";
    cout << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
                  // дефиниция на функция за обработка
int move(double d[N][M], double x[])
{ int i, j, k=0;
  for (i=0; i<N; i++)
    for (j=0; j<M; j++)
       if (d[i][j] > 0)
         {   x[k] = d[i][j];
             k++;
         }
  return k;           // връща резултат броя на числата
}
```

Идея:

При работа с входен масив (едномерен или двумерен), за който размерностите се въвеждат от клавиатурата, то на подпрограмата трябва да се добавят и формални параметри, предаващи тези размери. Тогава управляващите променливи в цикъл **i**, **j** се изменят не до константните **M**, **N**, а до въведените от ползващия програмата.

Задача f_4: Масивите **A(M)** и **B(N)** съдържат произволни въвеждани целочислени стойности. За всеки масив да се намери броят на елементите, които са кратни на 5.

Решение: За да се избегнат повтарящи се алгоритмични действия по въвеждане, извеждане и обработка на двата масива, решението на задачата се реализира с 3 подпрограми – **vhod** за въвеждане, **izhod** за извеждане и **obrab** за обработка на масив. Всяка от тези подпрограми се извиква по два пъти – веднъж за масива A и веднъж за масива B.

Формални параметри и в трите подпрограми трябва да са обработвания едномерен масив и броят на елементите му. Дефинирането на формален параметър за име на едномерен масив в език C/C++ е като указател ***x** или чрез **x[]**. И в двата варианта при извикването на подпрограмата фактически параметър е името на съответния едномерен масив.

Подпрограмата **vhod** е за въвеждане на стойности за елементите на едномерен масив **x** и броят на елементите му **k** (максимална размерност 20 елемента). Използват се два формални параметъра – **x** и **k**, които трябва да се връщат като резултат в главната програма. За да е изходен параметър, броят **k** трябва да се дефинира като указател ***k** (тогава при извикването му се подава адрес) или като псевдоним **&k**. Тук е използван вторият начин и затова в операторите **k** се използва с името си. Подпрограмата се извиква със съответните фактически параметри за име на обработвания масив и брой на елементите му, например **vhod(a, m)** и **vhod(b, n)**.

Подпрограмата **izhod** е за извеждане на стойностите на елементите на едномерен масив **x** с произволна размерност **k**. Използват се два формални параметъра – **x** и **k**, чрез които се подават входни данни, необходими за нейната работа. При извикване на подпрограмата с фактически параметри **izhod(a, m)** и **izhod(b, n)** се извеждат на екрана съответно елементите на масивите **a** и **b**.

Подпрограмата **obrab** е за обработване на едномерен масив **x** с **k** на брой елемента, с цел намиране на броя на елементите, които се делят на 5. Използват се три формални параметъра – **x**, **k** и **br**. Чрез **x** и **k** се подават входни данни за подпрограмата. Формалният параметър **br** е за пресметнатия резултат и като изходен параметър той е дефиниран като препратка **&br**.

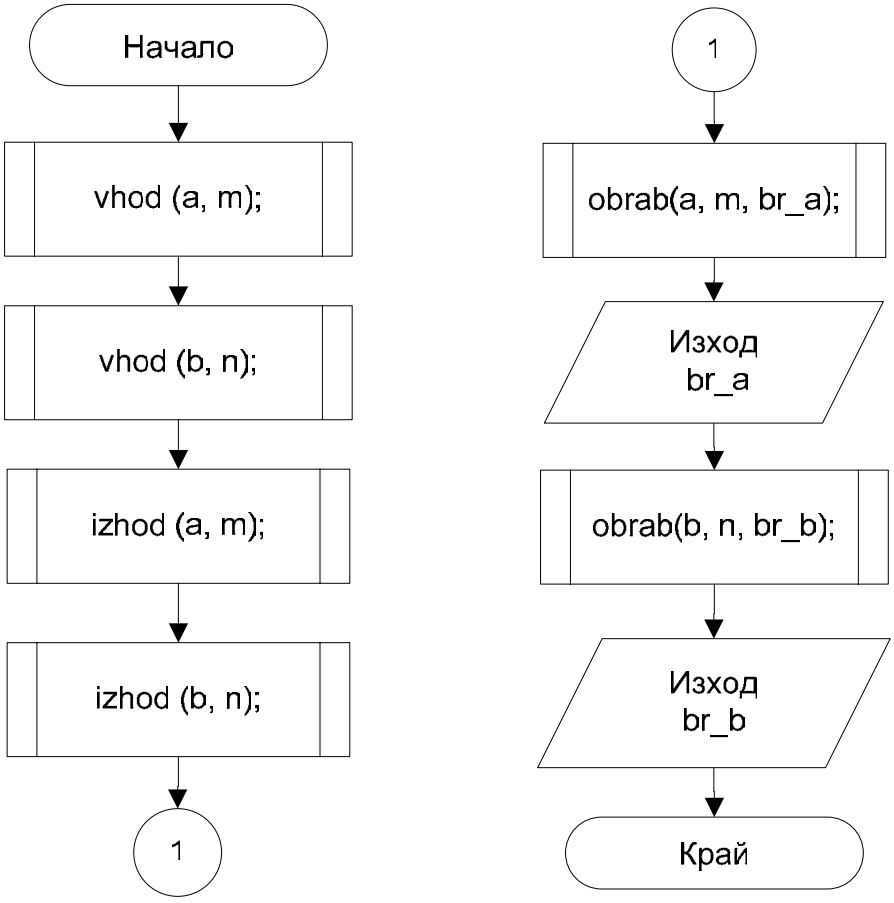
В програмата се използват и следните величини:

NM – константа за максимален брой елементи в масивите;

i – текущ индекс на елемент на масив и управляваща променлива за цикъл;

br_a, br_b – променливи за брой на кратните на 5 числа, резултати от двата масива в главната програма.

Блок-схемата на алгоритъма е представена в обобщен вид. Показана е само главната програма, в която се посочва мястото и последователността на извикваните подпрограми. Действията и обработката на данните, която извършват отделните подпрограми, са познати и вече са разгледани в предишни примери.



Програма на C++

```
#include <cstdlib>
#include <iostream>

using namespace std;

const int NM=20;      // глобална константа за размер
                      // прототипи на функциите:
void vhod (int x[], int &k);
void izhod (int x[], int k);
void obrab (int x[], int k, int &br);
```

```
int main(int argc, char *argv[])
{   int a[NM], b[NM];
    int n, m, br_a, br_b;
    vhod(a, m);
    vhod(b, n);
    izhod(a, m);
    izhod(b, n);
    obrab(a, m, br_a);
    cout << "Брой кратни на 5 в масив A = " << br_a <<
          endl;
    obrab(b, n, br_b);
    cout << "Брой кратни на 5 в масив B = " << br_b <<
          endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}

void vhod(int x[], int &k)      // функция за въвеждане
{ int i;
  do
  { cout << "Въведи брой на елементите =";
    cin >> k;
  } while (k<1);
  cout << "Въведи масив с " << k << " числа: \n";
  for (i=0; i<k; i++)
     cin >> x[i];
}

void izhod(int x[], int k)      // функция за извеждане
{ int i;
  cout << "\n Масив с " << k << " числа: \n";
  for (i=0; i<k; i++)
     cout << x[i] << "  ";
  cout << endl;
}
                                // функция за обработка
void obrab(int x[], int k, int &br)
{ int i;
  br=0;
  for (i=0; i<k; i++)
     if (x[i]%5 == 0)
        br++;
}
```

Задачи за самостоятелна работа

1. В масива z(30) да се намерят произведенията на ненулевите стойности, които са разположени в диапазона на 1-15 и 16-30 елемент. Програмата да се реализира с подпрограма за намиране на произведение на елементи от масив в зададен диапазон на индекси.
2. В целочисления масив c(20) да се намери броят на нечетните елементи, стоящи на места 1-10 и от 11-20. Пресмятането на броя на нечетните елементи от масив в зададен диапазон на индекси, да е в подпрограма.
3. В масива b(25) да се намерят средноаритметичното на числата, по-големи от 5 и средноаритметичното на числата, по-големи от 50. Програмата да се реализира с подпрограма за намиране на средноаритметична стойност на елементите от масив, които са по-големи от дадена стойност, зададена като входен параметър.
4. В масива a(20) да се намери броят на числата, които имат стойности в интервала [-5,5] и броят на тези в интервала [10,100]. Подпрограмата да е за намиране на брой на елементите от масив, които са в определен интервал с граници, зададени като входни параметри.
5. В масива p(15) да се намери сумата на числата, по-големи от 8, а в масива q(20) – сумата на числата, по-големи от -8. Да се реализира подпрограма за сумиране на елементи от масив, по-големи от определена стойност. Стойността и масива са входни параметри.
6. Да се намери най-големия елемент в масива d(N), където броят се въвежда и е N<30. Програмата да се реализира с подпрограма за намиране на максимален елемент в едномерен масив.
7. Да се намерят най-малките елементи в масивите s(10) t(15). Програмата да се реализира с подпрограма за намиране на минимален елемент в едномерен масив. Входни параметри са масивът и броят на елементите му.
8. От масивите x(20) и y(15) да се прехвърлят в масиви x1 и y1 всички ненулеви елементи. Да се реализира подпрограма за прехвърляне на елементи от едномерен масив в друг едномерен масив. Параметри са входния масив и броя на елементите му, изходния масив и като резултат – броят на елементите в него.