4. Алгоритми и програми за работа с едномерни масиви

Задача d_1: Да се състави програма, която да намира средната стойност на елементите на масив, който се състои от 6 елемента от реален тип.

Решение: Първо трябва да се въведат стойности за елементите на масива. След това всички елементи се сумират. Средната стойност се получава като се раздели получената сума на броя на елементите, който е 6.

При конкретно зададен брой за елементите на масива са възможни различни начини на работа. Единият начин е да се работи с константната стойност 6, а другият е да се дефинира именувана константа с тази стойност, например **n**=6. Вторият начин е по-добър, ако се налага програмата да се модифицира за друг брой елементи на масива. Тогава може да се промени само стойността на константата **n** и не се налага да се правят други промени в програмата.

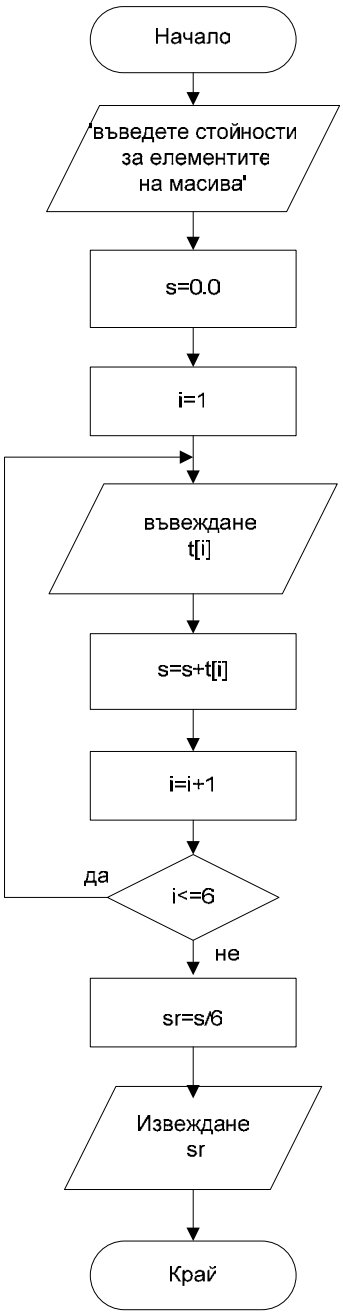
Обработката на елементите на масива е като се организира цикъл с управляваща променлива – брояч **i**, който се променя за индексите от 1 до **n** (или от 1 до 6).

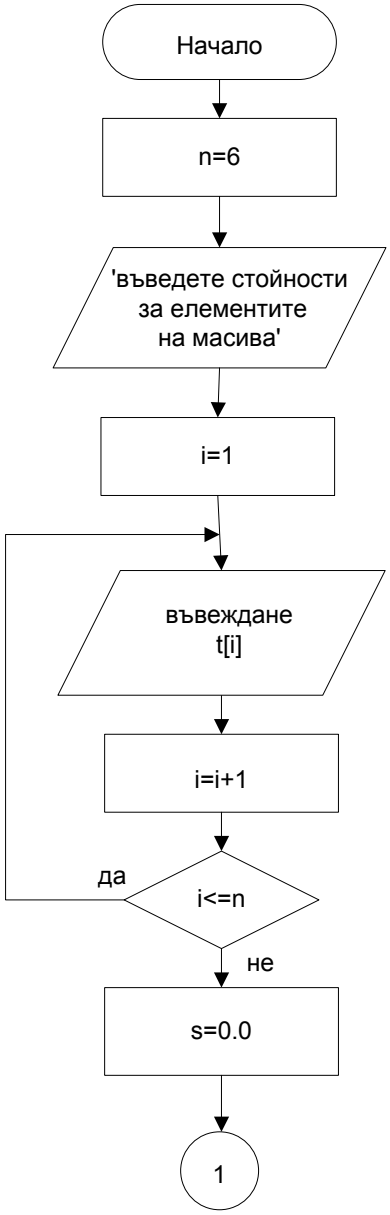
Важно: При масивите в език С и С++ първият елемент е с индекс 0. Затова вместо от 1 до 6, в програмата на С/С++ индексът **i** трябва да се изменя от 0 до 5, защото в паметта елемент с индекс 6 няма. Получава се разлика между индекса на елемента в програмата и поредния му номер. Освен в програмата, това може се отрази и в извежданите за потребителя съобщения и резултати. За да се виждат на екрана поредни номера като 1, 2, 3... може вместо индекс **i** да се извежда стойността **i+1**.

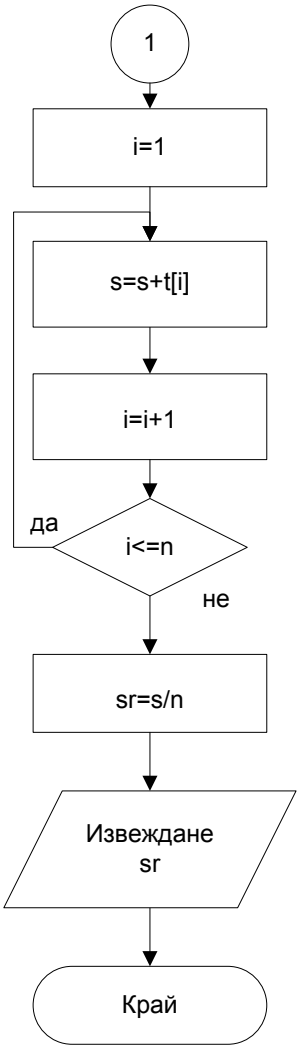
Варианти на решение: В първия алгоритъм е реализиран един цикъл, в който са включени и въвеждането и сумирането на елементите. Този вариант изглежда по-кратък, но за програми с по-сложна обработка може да е неприложим. Вторият показан вариант е с отделен цикъл за въвеждане на стойностите и отделен цикъл за обработка на масива за намиране на сумата на елементите му. Във втория алгоритъм, за брой на елементите на масива се използва именувана константа **n=6**.

Използват се следните променливи:

t – име на масива;
s – сума на елементите на масива;
sr – средна стойност;
i – управляваща променлива за цикъл и индекс на масива;
n – именувана константа за брой на елементите на масива







1) **Програма на Pascal** (по алгоритъм 1)

```
program d_1_1;
var i:integer;
    s,sr:real;
    t:array [1..6] of real;  {декларирane на масив}
begin
  s:=0.0;                    {начална стойност за сумата}
  writeln('въведете стойности за масива');
  for i:=1 to 6 do           {организира се цикъл}
  begin
    write ('t[',i,']=');
    readln(t[i]);  {въвежда стойност за текущ елемент}
    s:=s+t[i];     {елемента се добавя към сумата}
  end;             {край на цикъла}
```

```
  sr:=s/6;       {изчислява се средната стойност}
  writeln ('средната стойност е sr=',sr)
end.
```

2) **Програма на C** (по алгоритъм 2)

```
#include <stdio.h>
#include <stdlib.h>
#define  n  6
          /* дефинира се именувана константа  n=6  */
int main(int argc, char *argv[])
{   float t[n];          // дефиниране на масив  t[6]
    int i;               // индекс и брояч
    double s, sr;
    printf ("Въведи масив с %d елемента: \n", n);
    for (i=0; i<n; i++)   // цикъл с брояч за въвеждане
    {   printf ("t[%d]= ", i+1);
        scanf ("%f", &t[i]);   // въвеждане на елемент
    }
    s=0.0;
    for (i=0; i<n; i++)  // цикъл с брояч за обработка
        s += t[i];           // сумиране на елементите
    sr = s/n;
    printf ("Средна стойност = %lf \n", sr);
    system("PAUSE");
    return 0;
}
```

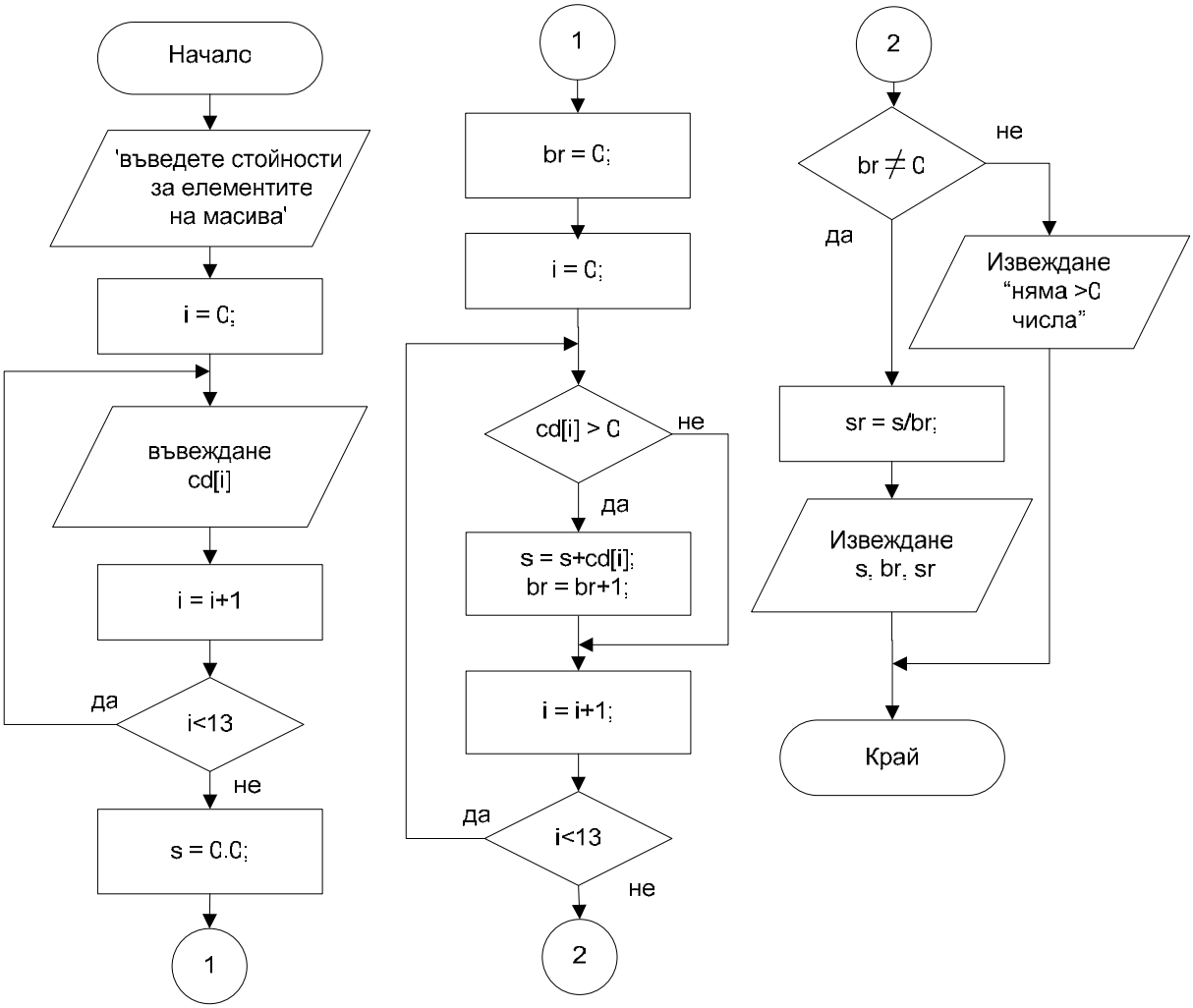
3) **Програма на C++** (по алгоритъм 2)

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

int main(int argc, char *argv[])
{   const int n=6;       // именувана константа
    float t[n];          // дефиниране на масив
    int i;
    double s, sr;
    cout << "Въведи масив с " << n << " елемента: \n";
    for (i=0; i<n; i++)    // цикъл с брояч i
    {   cout << "t[" << i+1 << "]=";
        cin >> t[i];       // въвеждане на елемент
    }
    s=0.0;
```

```
    for (i=0; i<n; i++)     // цикъл с брояч i
        s += t[i];          // сумиране на елементите
    sr = s/n;
    cout << "Средна стойност = " << sr << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Задача d_2: Да се въведат произволни реални стойности за елементите на масив CD, състоящ се от 13 елемента. Да се намери средната стойност на положителните елементи на масива. Да се изведат всички получени резултати.



Решение: За масива се въвеждат произволни стойности. При обработката е необходимо всички елементи на масива да се проверят дали отговарят на условието да са положителни. Задачата се свежда до намиране на сумата и броя на положителните елементи на масива, от които може да се пресметне средната стойност.

За да се избегне опит за деление на 0, изчислението за средна стойност е възможно само, ако има поне един положителен елемент. Това налага да се направи проверка дали има резултати и полученият брой е различен от 0.

Използват се следните променливи:

cd – едномерен масив;
i – управляваща променлива на цикъла и текущ индекс на елемент от масива;
br – брой на положителните елементи;
s – сума на положителните елементи;
sr – средна стойност на положителните елементи.

Програма на C++

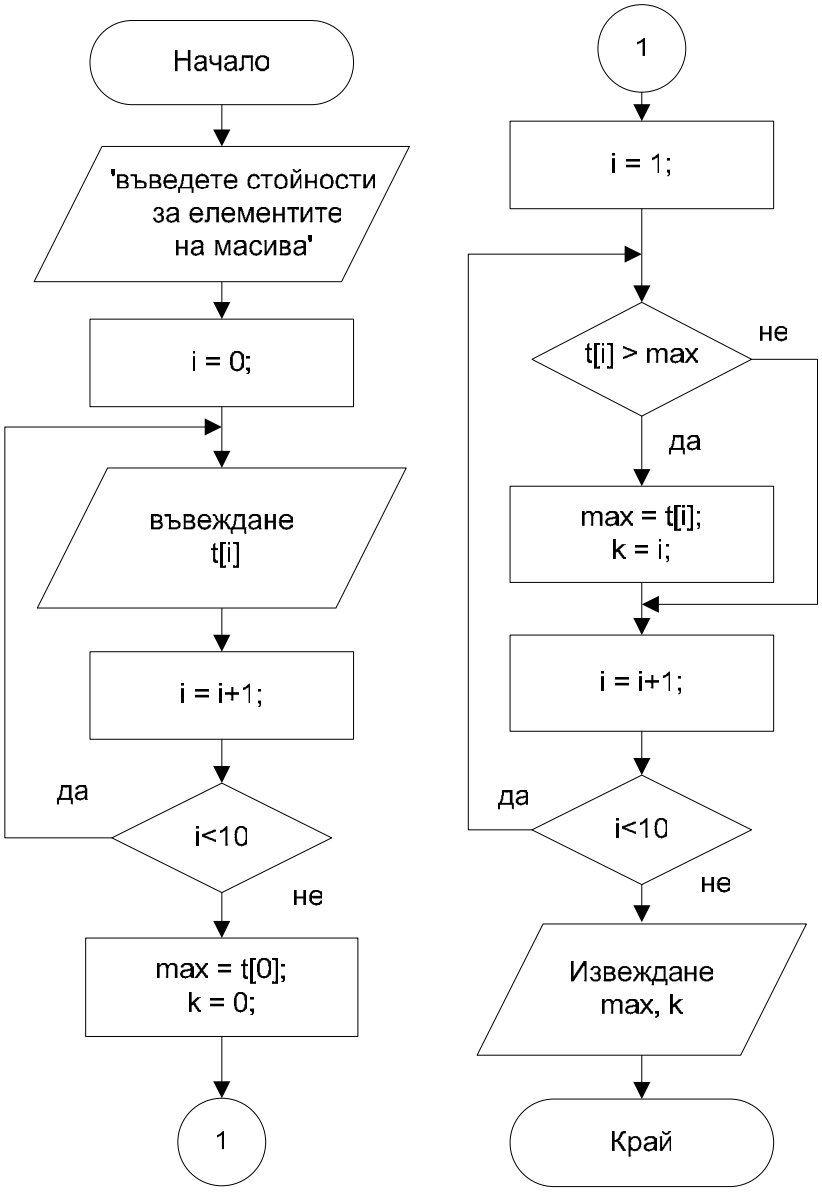
```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{   float cd[13];          // масив
    int i, br;
    double  s, sr;
    cout << "Въведи масив с 13 елемента: \n";
    for (i=0; i<13; i++)      // цикъл с брояч i
    {   cout << "cd[" << i+1 << "]=";
        cin >> cd[i];         // въвеждане на елемент
    }
    s=0.0;
    br=0;
    for (i=0; i<13; i++)      // цикъл с брояч i
      if(cd[i]>0)
      {  s += cd[i];           // сумиране на елементите
         br++;                 // преброяване +1
      }
    if (br!=0)        // проверка има ли резултати
    { sr = s/br;
      cout << "Сума на положителните = " << s << endl;
      cout << "Брой на положителните = " << br << endl;
      cout << "Средна стойност = " << sr << endl;
    }
    else
       cout << "Няма положителни числа. \n";
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Задача d_3: За едномерен масив **t,** който съдържа 10 реални елемента, да се намери стойността на максималния елемент и неговия пореден номер.

Решение: За решаването на тази задача трябва да се въведат две помощни променливи. Променливата **max** трябва да получи стойността на максималният елемент, а втората променлива **k** – индексът на този елемент от масива.

Първо променливите **max** и **k** трябва да получат подходящи начални стойности, най-добре: стойността и индекса на първия елемент от масива. След това стойността **max** се сравнява последователно с всички елементи на масива. Ако сравняването открие елемент от масива, който е по-голям от текущата стойност на **max,** то тогава **max** получава нова стойност - стойността на този елемент, а променливата за индекса **k** – получава индекса му.

Така когато се обработят всички елементи на масива, стойността на **max** е равна на стойността на най-големия елемент на масива, а стойността на **k** е индекса му.



Използвани променливи:

t – масив;
i – текущ индекс за елемент на масива и управляваща променлива за цикъла;
max – максимален елемент;
k – индекс на максималния елемент.

1) **Програма на C**

```
#include <stdio.h>
int main(int argc, char *argv[])
{   float t[10], max;
    int i, k;
    printf ("Въведи 10 числа: \n");
    for (i=0; i<10; i++)      // въвеждане на масива
    {   printf ("t[%d]= ", i+1);
        scanf ("%f", &t[i]);
    }
    max = t[0];       // начални стойности за max и k
    k = 0;
    for (i=1; i<10; i++)       // обработка на масива
        if (t[i]>max)
        {  max = t[i];     // нови стойности за max и k
           k = i;
           }
    printf (" Максимална стойност = %.2f \n", max);
    printf (" Пореден номер = %d \n", k+1);
  system("PAUSE");
  return 0;
}
```

2) **Програма на C++**

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{   double t[10], max;
    int i, k;
    cout << "Въведи 10 числа: \n";
    for (i=0; i<10; i++)      // въвеждане на масива
        cin >> t[i];
    max = t[0];       // начални стойности за max и k
    k = 0;
    for (i=1; i<10; i++)       // обработка на масива
        if (t[i]>max)     {    max = t[i];
                               k = i;
                          }
    cout << " Максимална стойност = " << max << endl;
    cout << " Пореден номер = " << k+1 << endl;
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Задача d_4: За едномерен масив, който съдържа най-много 20 елемента, да се изведат елементите с четен индекс.

Решение: Представени са два варианта на решение.

При първия вариант четният индекс е резултат от нарастване на текущия индекс с 2.

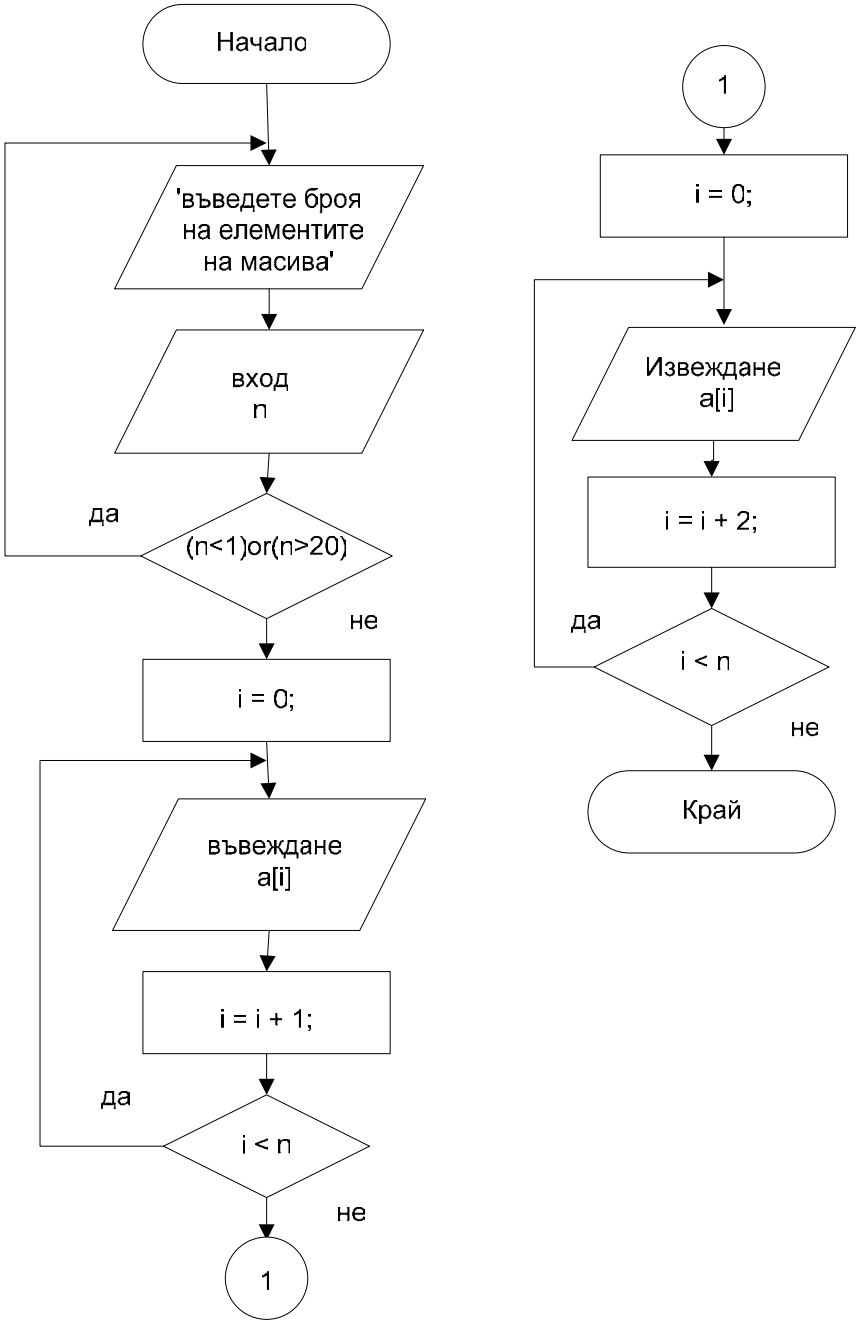
При втория вариант се проверява дали текущият индекс е четен, като условието проверява какъв е остатъкът от делението на 2 (т.нар. деление по модул mod).

Използват се следните променливи:

a – име на масива;

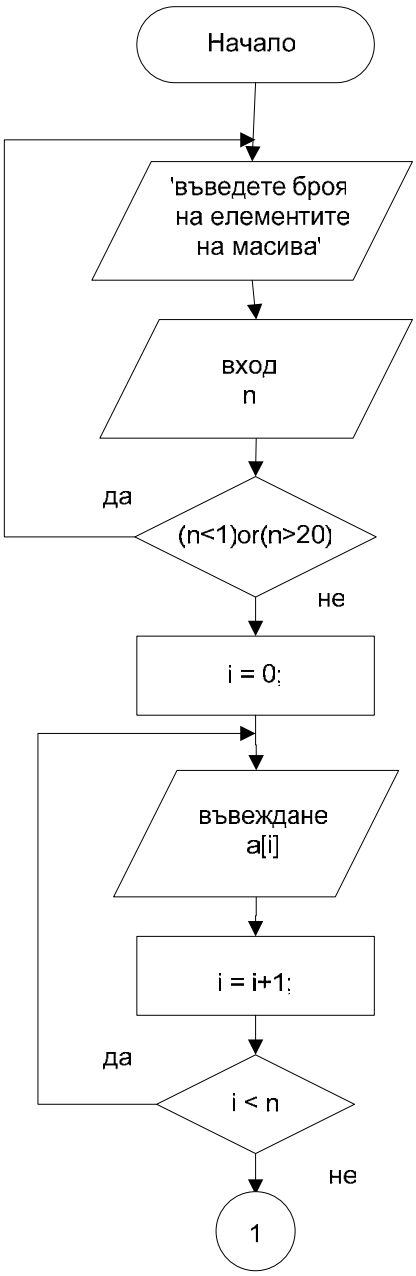
i – текущ индекс за елемент на масив и управляваща променлива за цикъла;

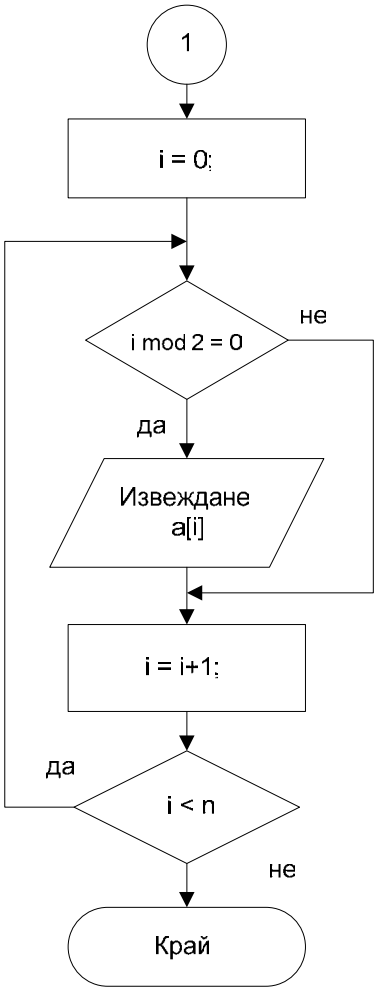
n – брой на елементите на масива.



1) **Програма на C++** (алгоритъм 1)

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{   double  a[20];
    int  i, n;
    do                  // въвеждане на броя n
    {   cout << "Въведи брой n = ";
        cin >> n;
    } while (n<1 || n>20);
    cout << "Въведи " << n << " числа: \n";
    for (i=0; i<n; i++)      // въвеждане на масива
        cin >> a[i];
```





```
    cout << "\nЕлементи на четни индекси: \n";
    for (i=0; i<n; i+=2)      // броячът нараства с +2
            cout << a[i] << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

2) **Програма на C++** (алгоритъм 2)

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{   double a[20];
    int i, n;
    do                // въвеждане на броя n
    {   cout << "Въведи брой n = ";
        cin >> n;
    } while (n<1 || n>20);
    cout << "Въведи " << n << " числа: \n";
    for (i=0; i<n; i++)     // въвеждане на масива
        cin >> a[i];
    cout <<  "\nЕлементи на четни индекси: \n";
    for (i=0; i<n; i++)
        if (i%2 == 0)     // условие четно ли е i
            cout << a[i] << "  ";
    cout <<  endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Идея: Такава обработка, при която откриването на четни или нечетни стойности е чрез проверка на стойността на остатъка от делението на 2, е подходяща да се реализира и ако в програмата трябва да се проверяват за кратност не индексите на елементите, а самите елементи на масива, които са произволни обработвани числа.

Какъв трябва да е използвания тип на данните?

Остатък от деление се изчислява само с целочислени данни. А ако числата са от реален тип, те могат да се преобразуват към целочислени чрез явно преобразуване.

Например:

```
double  num1 = 6.78;
int  num2 = (int)num1;      // Резултатът е  num2 = 6
num2 = (int)(num1/2);       // Резултатът е  num2 = 3
```

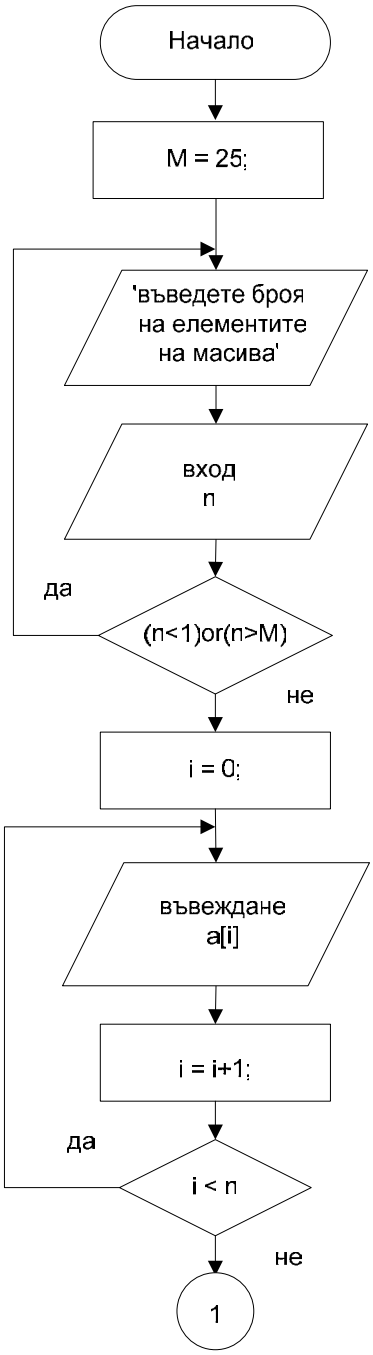
Задача d_5: От едномерен масив **a,** който съдържа най-много 25 на брой реални стойности, да се формира нов масив **b** по следното правило:

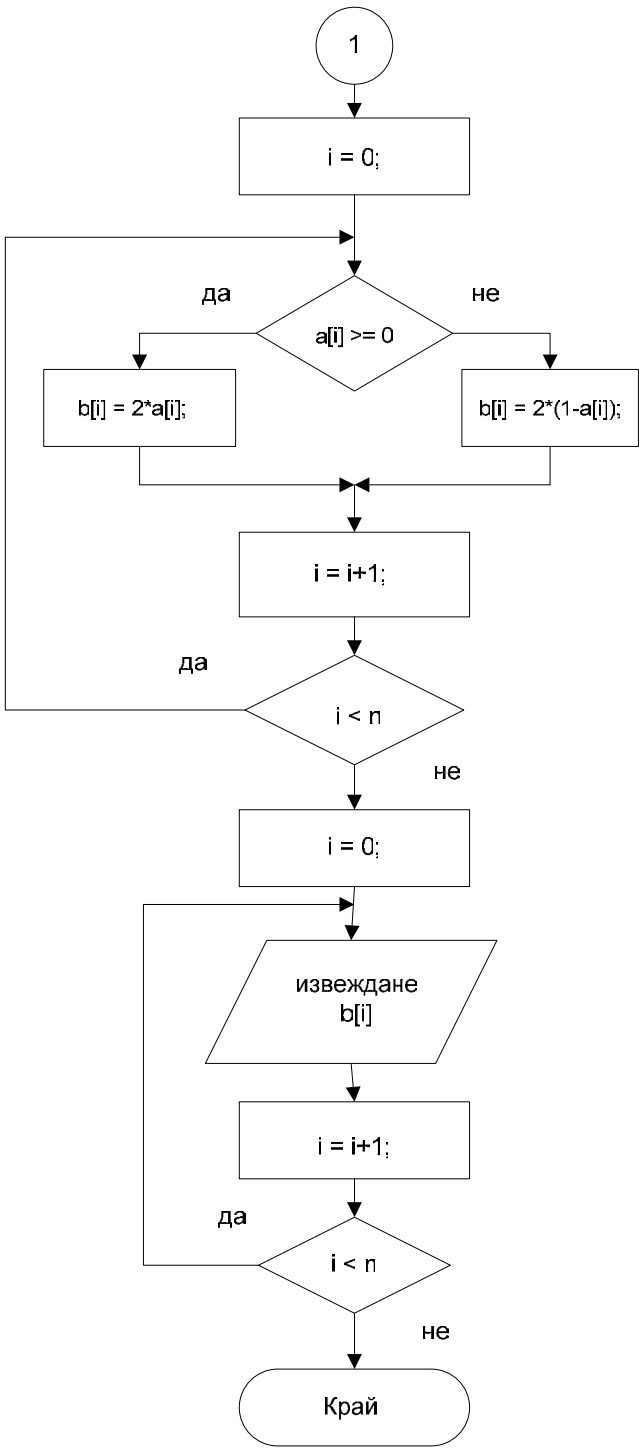
$b_i = 2a_i$ за $a_i \geq 0$

$b_i = 2(1-a_i)$ за $a_i < 0$.

Да се изведе новополучения масив **b.**

Решение: В този тип задачи на всеки елемент от входния масив **a** съответства елемент от новоформирания масив **b** със същия индекс. Двата масива имат еднаква размерност.





Въвеждат се стойности за елементите на масив **a**. Масивът **a** се обработва, като всеки елемент се проверява на кое условие отговаря и в зависимост от това се изчислява съответния елемент от масива **b** според зададеното условие.

Добър начин на работа на програмата е въвеждането и извеждането да са поотделно, т.е. първо да се въвеждат входните данни, а след това на екрана да се извеждат получените резултати. В представения вариант въвеждането на стойности, формирането на новия масив и извеждането на новия масив се извършва с три отделни цикъла.

Използват се следните величини:

a – входен масив;
b – новополучен масив;
i – текущ индекс на елемент от масивите и управляваща променлива за цикъл;
n – брой на елементите на масивите.

Програма на C++

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{    const int M=25;       // константа за брой елементи
     double a[M], b[M];
     int i, n;
     do
     {    cout << "Въведи брой n = ";
          cin >> n;
     } while (n<1 || n>M);
     cout << "Въведи " << n << " реални числа: \n";
     for (i=0; i<n; i++)      // въвеждане на масив
     {   cout << "a[" << i+1 << "]=";
         cin >> a[i];
     }
     for (i=0; i<n; i++)      // обработка
          if (a[i]>=0)
              b[i] = 2*a[i];
          else
              b[i] = 2*(1-a[i]);
     cout <<  "\n Нов масив: \n";     // извеждане
     for (i=0; i<n; i++)
          cout << "b[" << i+1 << "]=" << b[i] << endl;
     system("PAUSE");
     return EXIT_SUCCESS;
}
```

Задача d_6: От едномерен масив, който съдържа 15 цели стойности, да се отделят в нов масив само тези стойности, които са по-малки от 0 или по-големи от 100. Да се изведат входния и новополучения масив.

Решение: Това е тип задачи, при които от един масив се отделят в друг масив само елементи, които отговарят на зададено условие. Важен момент при решаването на задачата е да се определи размерността на новофор-мирания масив. Ако всички елементи на входния масив отговарят на зададеното условие, то новият масив ще има толкова елемента, колкото входния, т.е двата масива ще имат еднаква размерност. Но новият масив може и да не съществува, ако никой от елементите на входния масив не отговаря на зададеното условие.

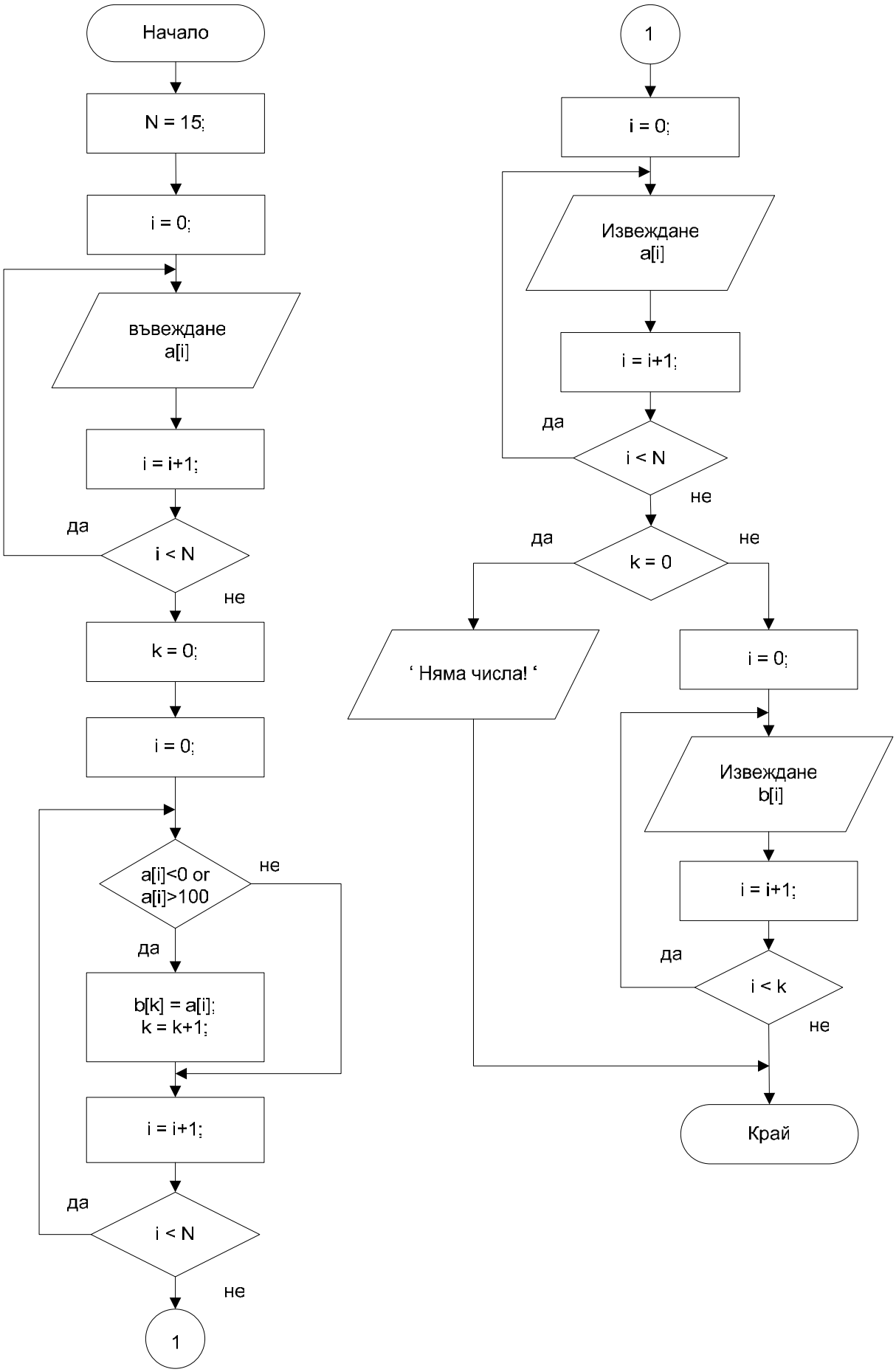
При този тип задачи индексите на двата масива не съвпадат и е необходимо да се използва допълнителна променлива за текущ индекс на новополучения масив. Работата с тази променлива за индекс може да е по два начина – нарастването на променливата е преди или след съхраняването на стойност за елемент на масива. И двата начина са приложими в програмите на Pascal и C/C++.

Първоначално индексът за новия масив получава подходяща начална стойност за първия елемент на масива, обикновено 0 при език C/C++ или 1 при Pascal. В циклич-на структура входният масив се обработва по елементи и когато се появи елемент, който удовлетворява зададеното условие, тогава стойността му се присвоява на текущия елемент от новия масив и индексът на новия масив се увеличава с 1, за да е подготвен за евентуален нов елемент от масива. След приключване на обработката на входния масив, стойността на индекса на новия масив показва и какъв е броят на елементите в новия масив.

При Паскал първоначално индексът също може да е подготвен с начална стойност 0. При обработката според задачата, той първо нараства и така формира пореден елемент от новия масив, в който се записва текущата стойност, удовлетворяваща условието.

Използват се следните променливи величини:

a – входен масив;
b – новополучен масив;
i – индекс на масива и управляваща променлива за цикъл;
k – помощна променлива за текущ индекс на новополучения масив и брой на елементите му.


Начало
N = 15;
i = 0;
въвеждане
a[i]
i = i+1;
да
i < N
не
k = 0;
i = 0;
a[i]<0 or
a[i]>100
не
да
b[k] = a[i];
k = k+1;
i = i+1;
да
i < N
не
1
1
i = 0;
Извеждане
a[i]
i = i+1;
да
i < N
не
да
k = 0
не
‘ Няма числа! ‘
i = 0;
Извеждане
b[i]
i = i+1;
да
i < k
не
Край

Програма на C++

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

using namespace std;

int main(int argc, char *argv[])
{
    const int  N=15;      // константа за брой елементи
    int  a[N], b[N];      // масиви
    int  i, k;            // броячи за двата масива
// въвеждане на входния масив
    cout << "Въведи " << n << " цели числа: \n";
    for (i=0; i<N; i++)
    {   cout << "a[" << i+1 << "]=";
        cin >> a[i];
    }
// обработка за създаване на новия масив
    k=0;
    for (i=0; i<N; i++)
        if (a[i]<0 || a[i]>100)
        {   b[k] = a[i];
            k++;
        }
// извеждане на входните данни
    cout <<  "\n Входен масив: \n";
    for (i=0; i<N; i++)
        cout << "a[" << i+1 << "]=" << a[i] << endl;
// извеждане на резултати
    if (k==0)             // ако няма нов масив
        cout << "\n Няма числа <0 или >100! \n";
    else
    {
       cout <<  "\n Нов масив: \n";
       for (i=0; i<k; i++)
         cout << "b[" << i+1 << "]=" << b[i] << endl;
    }
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Задачи за самостоятелна работа

1. В масива b(30) да се намери произведението на числата, които са по-големи или равни на 5.
2. В масива c(40) да се намери броят на ненулевите елементи, стоящи на нечетни индекси.
3. В масива d(45) да се намери броят на числата, по-големи от -5.
4. В масива e(30) да се намери произведението на ненулевите елементи, с изключение на първия и последния елемент.
5. В масива f(35) да се намери броят на ”единиците”, стоящи на четни индекси.
6. В масива a(30) да се намери сумата на отрицателните числа, стоящи на нечетни индекси.
7. В масива b(35) да се намери броят и произведението на числата, по-големи или равни на 2.
8. В масива c(40) да се намери броят на числата, попадащи в интервала [a, b] и сумата на числата, които са извън интервала [a, b].
9. В масива x(N) да се намери сумата и произведението на всички ненулеви елементи. Броят на елементите се въвежда и е 10<N<50.
10. В масива y(M) да се намери броят на нулевите елементи, стоящи на индекси, кратни на 4 (y0, y4, y8, ...). Броят на елементите се въвежда и е 20<M<60.
11. В целочисления масив z(K) намери сумата на числата, чиято стойност е кратна на 5. Броят на елементите се въвежда и е 10<K<40.
12. В масива p(N) да се намери най-малкият елемент и неговия пореден номер. Броят на елементите се въвежда и е 5<N<30.
13. От масива x(M) да се прехвърлят в нов масив y всички ненулеви елементи. Броят на елементите се въвежда и е 1<M<20.
14. От целочисления масив y(B) да се прехвърлят в нов масив z всички положителни нечетни елементи. Броят на елементите се въвежда и е 1<B<20.
15. От масива z40 да се прехвърлят в нов масив g всички елементи, без максималния елемент и равните на него.