Алгоритми и свойства на алгоритмите

Един от най-важните етапи при проектирането на една програма е съставянето на нейният алгоритъм. За това какво са алгоритмите, какви общи свойства имат те и как се изпълняват ще поговорим в този урок.

Думата алгоритъм произлиза от името на един от най-големинте учени на Средна Азия Мухамада ибн Мус ал-Хорезми. Той е написал редица книги по аритметика и алгебра и често е наричан бащата на алгебрата, смята се че той поставя нейните основи.

Алгоритми и свойства на алгоритмите.

Използването на компютри за автоматизация на обработката на информация и управление в която и да е човешка дейност се основава на принципа на програмното управление. В основата на компютърното управление е програмата. Всяка програма се пише от човек, но преди това е необходимо да се опише пълно, ясно и еднозначно изчислителният процес. Това еднозначно описание на процеса представлява алгоритмът на действието и.

Под алгоритъм се разбира: **Точно упътване за решаване на един проблем или определен вид проблеми. Той се състои от крайна поредица от указания, които се изпълняват едно след друго и отчасти повтарят в определен ред.**

Между понятията алгоритъм и програма има частично съответствие — всеки алгоритъм може да се представи като програма, но не всяка програма е алгоритъм.

За да се определи дали дадено описание на действия е алгоритъм или не, трябва да се провери дали то отговаря на определени условия:

- ***Дискретност***, т.е. да се състои от отделни, разграничени по време една от друга стъпки, всяка от които се извършва за крайно време.
- ***Детерминираност (определеност)*** , т.е. резултатът от действията върху данните и посочването на следващата стъпка трябва да бъдат еднозначно определени от действията, извършени до текущия момент.
- ***Масовост***, т.е. процедурата трябва да свършва с резултат не за краен брой съчетания от входните данни, а за едно потенциално безкрайно множество от входни данни.
- ***Резултатност***, т.е. процедурата трябва винаги да дава резултат, ако входните данни принадлежат на даденото подмножество.

- ***Крайност на изпълнението,*** т.е. процедурата трябва да свършва с резултат за краен брой стъпки.

Пример за процедури, които не са алгоритми, са т. нар. зациклени процедури — последователности, образуващи затворен цикъл от действия без изход. Такива процедури никога не завършват.

В зависимост от своята структура алгоритмите могат да бъдат:

- **Линейни** – При тях последователността от действия винаги е една и съща. Всяка стъпка на алгоритъма има само една предходна и една следваща.
- **Разклонени** – Последователността от извършвани действия зависи от стойността на входните данни. Тези алгоритми съдържат команди, при които в зависимост от изпълнението или не на дадено условие, се определят следващите за изпълнение команди
- **Циклични** – Определена част от действията се повтаря многократно докато не се изпълни зададеното условие.

Най-често се срещат комбинираните алгоритми при които отделни части отговарят на линейни, разклонени или циклични.

Средства за описване на алгоритми.

За представяне на алгоритмите се използват различни езици: естествени, алгоритмични, параметрични, графични и др., които се характеризират с различна степен на формализация на описанието, удобство за запис и четене, възможност за обмен и т.н. Всяка програма е специфично представяне на алгоритъма, разбира се, като се пренебрегнат ограниченията, налагани от съответния език за програмиране.

а) Словесно описание

Словесното описание представлява набор от указания, в които чрез думи от някакъв естествен език са посочени действията, които трябва да бъдат извършени. Възможно е при описанието да се използват означения от съответната предметна област. Този начин на описание не е много популярен, тъй като за много думи от естествените езици съществува неоднозначно тълкуване.

б) описание чрез език за програмиране (псевдокод)

Алгоритмите се описват като се използва ограничено множество от конструкции на естествен език. Множеството от конструкции се разглежда като език за проектиране на алгоритми.

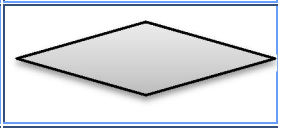
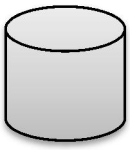
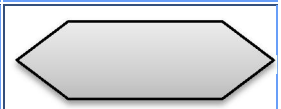
Пример: Алгоритъм на Евклид чрез псевдокод

```
Въведи A, B
Докато A ≠ B повтаряй
   Ако A>B, то A:=A-B, иначе B:=B-A;
Изведи A
Край.
```

в) Графично описание посредством блок-схеми;

Блок схемата е графично изразно средство, за описание на алгоритми, което дава възможност ясно да се покажат връзките между отделните инструкции.

Таблица 1: Стандартни символи използвани за описание на блокови алгоритми

Наименование	Символ	Изпълнявана функция
1. Изчислителен блок		Изпълнява изчислително действие или група от действия
2. Логически блок		Избор на направления за изпълнение на алгоритма в зависимости от условието
3. Блокове за въвеждане и извеждане на информация		Въвеждане или извеждане на данни в зависимост от физическия носител
		Извеждане на данни на печатащо устройство
		Ръчно въвеждане на данни
		Дисплей
		Четене и запис от хард диск
4. Начало/край		Начало или край на програмата или подпрограма
5. Подпрограма		Изпълнение на стандартна или потребителска подпрограма
6. Блок модификация		Изпълнение на действия за промяна на точката (пункта) на алгоритъма
7. Вътрешно-страничен съединител		Вътрешно страничен преход
8. Между-страничен съединител		Между-страничен преход (преминаване на нова страница)

Описанието чрез блок-схеми е удобно, тъй като се предлага визуална представа на логическите връзки между отделните действия в алгоритъма. Описанието се извършва от определени символи, наречени блокове, като всеки символ има точно определен смисъл. Блоковете имат вида на геометричните фигури: правоъгълник, ромб, успоредник. Последователността се задава чрез стрелки. Блок-схемата започва с начален блок, в който се записва указанието, с което започва алгоритъма. В таблица 1 са показани по-важните приети у нас по стандарт символи използвани в блоковите схеми (БДС 19103—78).

Етапи на пълното разработване на алгоритми и програми

Решението на дадена задача се разбива на няколко етапа:

- Постановка на задачата
- Формализация (математическа постановка)
- Избор (или разработка) на метод за решаване
- Разработване на алгоритъма
- Проверка на правилността на алгоритъма
- Съставяне на програма
- Настройка на програмата
- Обработка и анализ на получените резултати при работата на алгоритъма
- Съставяне на документацията

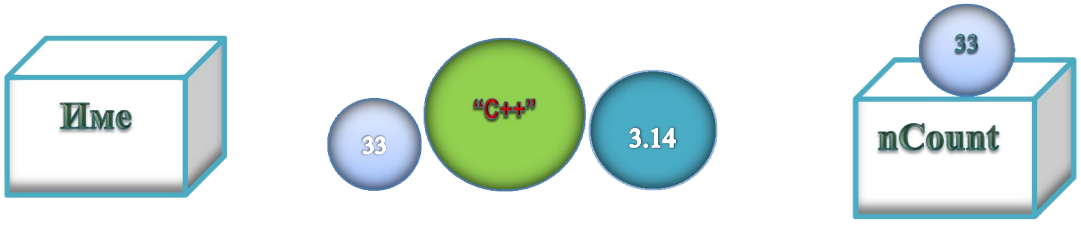
Основни видове алгоритми

Преди да разгледаме основните видове алгоритми нека се запознаем с някои от основните понятия при работа с алгоритми.

а) Величини - характеризират се с име и тип. Типът определя множеството от допустими стойности на величината. Стойността на величина в даден момент се нарича текуща стойност.

Величините се делят на константи и променливи:

- **Променлива** – Математическата представа за променлива е различна от понятието променлива при алгоритмите. В компютърните алгоритми, променливата означава елемент от паметта, чието съдържание може да се променя. На фиг. 1 имената на променливите са представени като кутии, те са отделени от своите стойности които са изобразени като топки. Резултата е показан на променливата с име **nCount** добива стойност 33.



Променлива | Стойности | Променлива nCount със стойност 33

Фиг.1 Абстрактно представяне на променливи

- **Константи** – тяхната стойност не се променя.

b) Изрази – за всеки тип величини са дефинирани различни операции, които се използват в изрази за изчисление.

В изразите величините участват със своите имена, а операциите се извършват върху стойностите им.

Линейни алгоритми

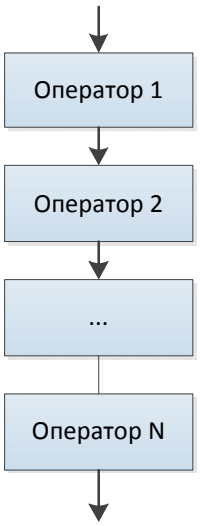
При тях последователността от действия винаги е една и съща. Всяка стъпка на алгоритъма има само една предходна и една следваща (фиг. 2).

Пример:

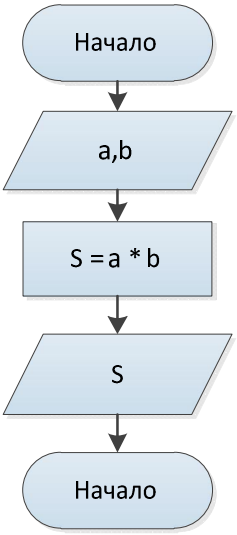
Нека да разгледаме алгоритъма за пресмятане лицето на правоъгълник по формулата $S = a * b$.

Стъпка 1: Начало.
Стъпка 2: Въведете и запомнете стойностите на a и b.
Стъпка 3: На променливата S присвояваме произведението на $a * b$
Стъпка 4: Изведете стойността на S като резултат.
Стъпка 5: Край.

Блоковата схема на алгоритъма е дадена на фиг. 3



Фиг. 2. Блокова схема на линеен алгоритъм



Фиг. 3. Блокова схема на алгоритъм за решаване на формулата $S = a * b$

Разклонени алгоритми

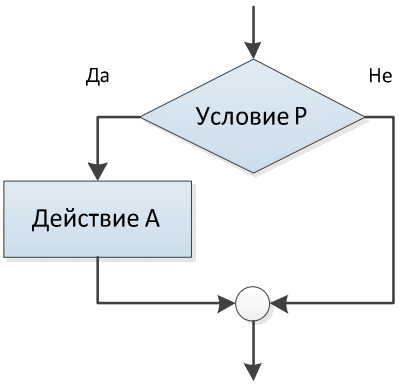
Много често се налага да избираме между няколко възможни варианта. Какво решение ще вземем обикновено зависи от някакво условие. Тоест при тях последователността от извършвани действия зависи от стойността на входните данни. Тези алгоритми съдържат команди, при които в зависимост от изпълнението или не на дадено условие, се определят следващите за изпълнение команди. Подобни условия се наричат логически.

Съществуват две форми на разклонен алгоритъм:

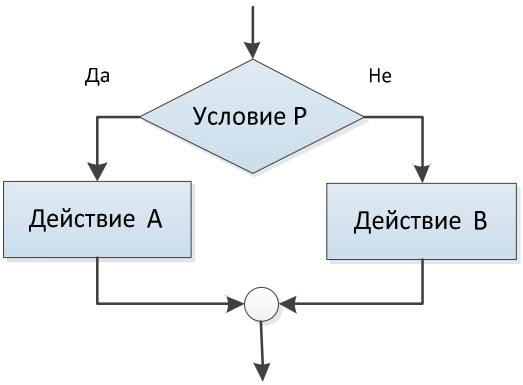
Кратка – Използва се тогава, когато е необходимо да се изпълни

действието А само при определено условие Р. Ако условието не е изпълнено, то командата не се изпълнява.

Пълна – Ако се изпълни условието Р, се извършва действие А, а ако не се изпълни действие В.



Фиг. 4. Условен оператор в непълен формат



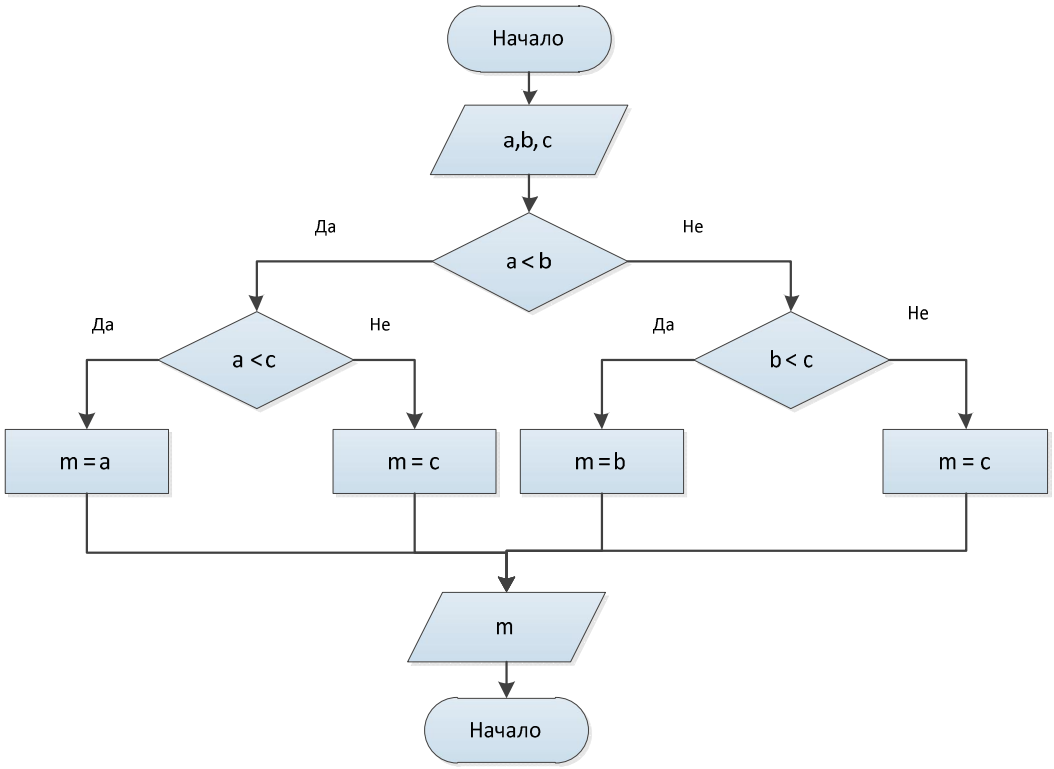
Фиг. 5. Условен оператор в пълен формат

Пример: Да се синтезира алгоритъм за намиране на най-малкото от три числа.

Така поставената задача може да бъде решена по два начина. При първият се изпълняват следните стъпки:

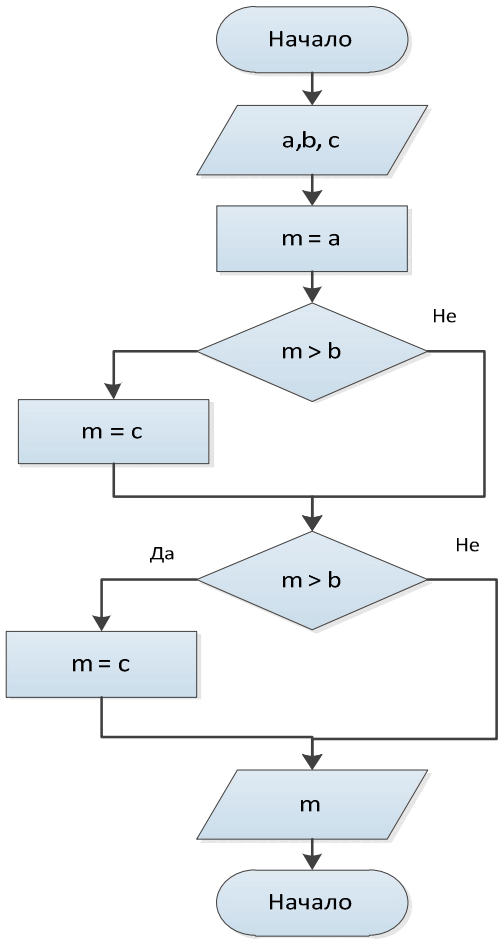
Стъпка 1: Начало.
Стъпка 2: Въведете и запомнете стойностите на a, b и c.
Стъпка 3: Проверете дали a < b
Стъпка 4: Ако a < b, проверете дали a<c
Стъпка 5: Ако a<c на m присвоете стойността на a, ако не е присвоете на m стойността на c
Стъпка 6: Ако a ≥ b, проверете дали b<c
Стъпка 7: Ако b<c на m присвоете стойността на b, ако не е присвоете на m стойността на c
Стъпка 8: Изведете стойността на m като резултат.
Стъпка 9: Край.

Алгоритмът е даден на фиг. 6



Фиг.6. Алгоритъм на програма за намиране на най-малкото от 3 числа

Втори вариант на алгоритъма е показан на фиг. 7. Опитайте се да запишете стъпките за изпълнението му.



Фиг.7. Алгоритъм на програма за намиране на най-малкото от 3 числа – втори вариант

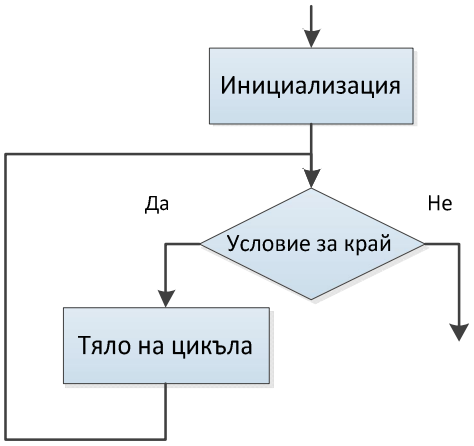
Циклични алгоритми

Много често се налага определени действия да бъдат повторени многократно докато не се изпълни зададеното условие. Групата от блокове, които са описани веднъж, но се изпълняват многократно се наричат ТЯЛО НА ЦИКЪЛА. При тях задължително има условен блок. А необходимо изискване е да се осигури край на цикличните повторения (излизане от цикъла). В тялото на цикъла, трябва да се променя стойността поне на 1 променлива участваща в условието за край. В противен случай ще се получи безкраен цикъл.

Циклите биват 3 вида:

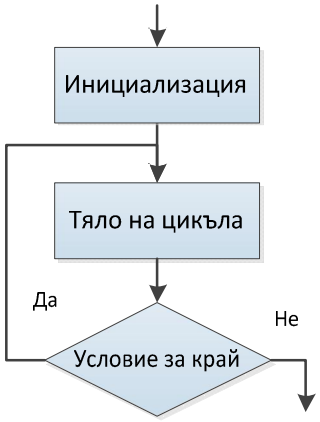
- Цикъл с предусловие
- Цикъл със следусловие (постусловие
- Цикъл с параметри

1. **Цикъл с предусловие** – при тях първо се проверява условието и ако то е изпълнено се изпълняват операциите от тялото на цикъла. Ако не е изпълнено още в началото е възможно тялото на цикъла да не се изпълни нито веднъж.



Фиг. 8 Цикъл с предусловие

2. **Цикъл със следусловие (постусловие)** – при тях първо се изпълняват операциите от тялото на цикъла и след това се проверява условието за продължаване на цикъла. Ако то е изпълнено се връща за повторно изпълнение на тялото, ако не е се излиза от цикъла.



Фиг. 9 Цикъл със следусловие

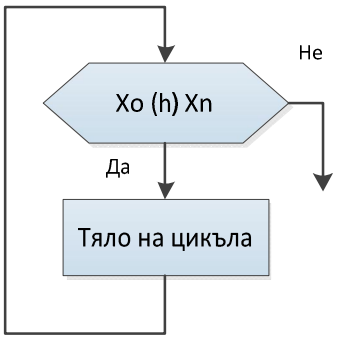
3. **Цикъл с параметри,** или цикъл с брояч или аритметичен цикъл- това е цикъл с предварително зададен брой повторения.

В блока за условие се задава закона за промяна на параметъра.

Xo - първоначалната стойност на бояча

h– (индексна променлива) брояч

Xn – последна стойност на брояча



Фиг. 10 Цикъл с параметри

При използването на цикли с броячи трябва да се спазват следните условия:

- Началната и крайната стойност на брояча трябва да са от един и същ тип.
- Забранява се в тялото на цикъла да се променя значението на началната стойност.
- Ако началната стойност на брояча е по-голяма от крайната стъпката на нарастване е отрицателна.
- След като излезете от тялото на цикъла индексната променлива е неопределена и неможе да се използва.

Всеки един цикъл има 4 задължителни елемента:

- Установяване на началното състояние (началните стойности);
- Тяло – група от оператори, които се изпълняват многократно;
- Условие за изход от цикъла;
- Обновяване на условието за излизане от цикъла (брояч).

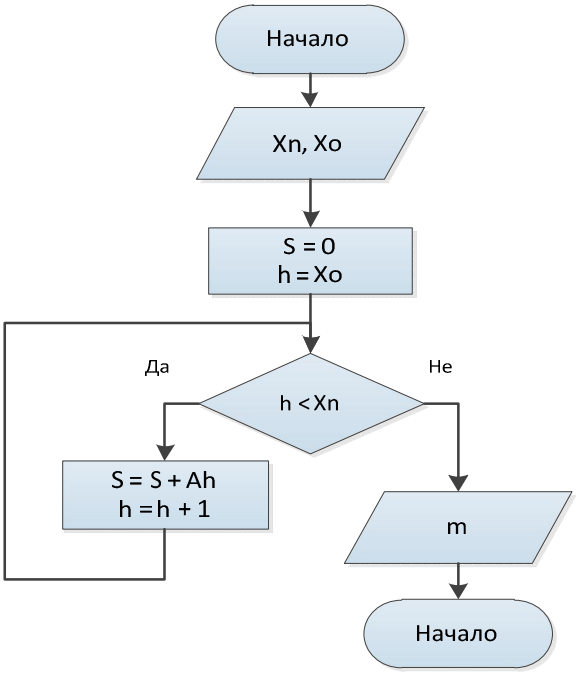
Пример:

Пример за цикъл е задачата за намиране сумата на редица от числа. Сумиране на голям брой числа се използва при обработката на различна статистическа информация или при анализа на различни динамични процеси.

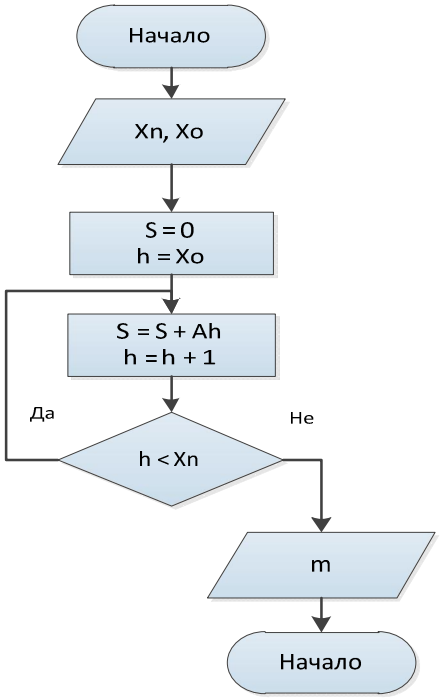
В примера индексната променлива ще бъде зададена като h. Началната стойност като Xo. Крайната с Xn, а сумата със S.

В алгоритъма първо се въвеждат стойностите на Xo и Xn. Задава се начална стойност 0 на S. На брояча h се задава стойността на Xo. Проверява се дали стойността на брояча е достигнала крайната стойност Xn. Ако не към стойността на S се добавя стойността на поредния елемент на масива (под масив в програмирането се разбира

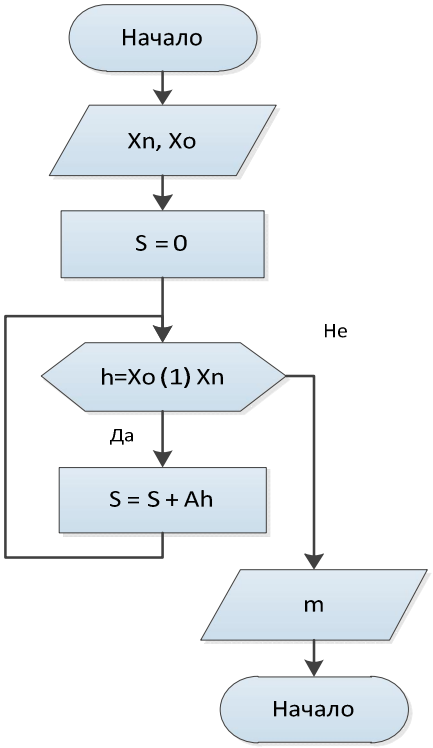
вектор ред от математиката).



Фиг.11. Алгоритъм на програма за намиране сумата на редица от числа реализирана с цикъл с пред условие



Фиг.12. Алгоритъм на програма за намиране сумата на редица от числа реализирана с цикъл със следусловие



Фиг.13. Алгоритъм на програма за намиране сумата на редица от числа реализирана с цикъл с параметри

Примери за алгоритми

За представяне на алгоритмите се използват различни езици: естествени, алгоритмични, параметрични, графични и др., които се характеризират с различна степен на формализация на описанието, удобство за запис и четене, възможност за обмен и т.н. Всяка програма е специфично представяне на алгоритъма, разбира се, като се пренебрегнат ограниченията, налагани от съответния език за програмиране.

Езикът на блоковите схеми е опростена разновидност на графичните езици. *Блоковата схема* е описание на алгоритъма, което се състои от върхове (възли) и съединяващи ги клонове. Клоновете са отбелязани със стрелки, които показват посоката на движение, т.е. реда на изпълнение на операциите.

Пример: Да се състави блоковата схема на алгоритъма за провеждане на телефонен разговор от кабина при известен номер.

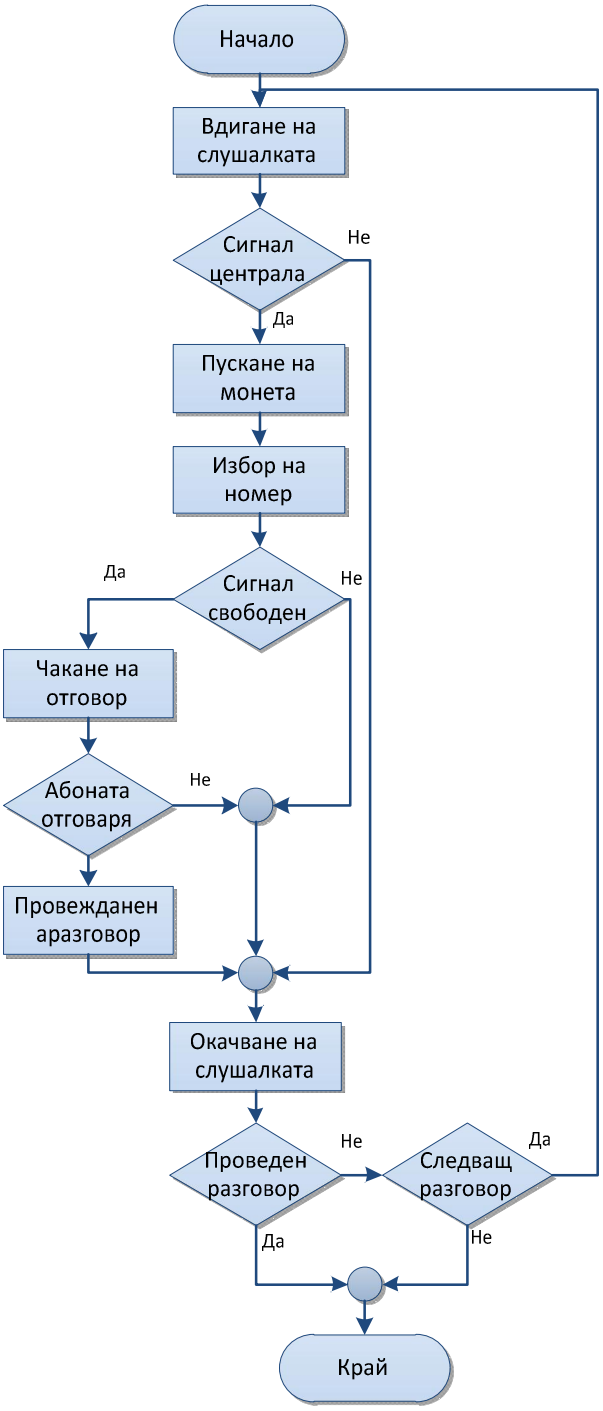
Не представлява трудност да се опише словесно последователността от действия, за да се свърже с даден абонат. В действителност е необходимо не само да се извършват съответните действия, а и да се анализират някои условия, в зависимост от които следва нашето действие. На естествен език алгоритмът може да бъде описан със следната последователност от стъпки:

- Вдигнете слушалката и ако чувате сигнал „централа", пуснете жетон и преминете към т.2. Ако няма сигнал или той е „заето", преминете към т.6.
- Изберете желания номер.
- Ако сигналът е „заето" или друг сигнал, преминете към т.6, а в противен случай изпълнете т.4.
- Изчакайте отговора на абоната. Ако абонатът отговаря, преминете към т.5, а в противен случай към т.6.
- Проведете разговора.
- Окачете слушалката.
- При проведен разговор преминете към т.9, а в противен случай към т.8.
- При следващ опит за връзка преминете към т.1, а в противен случай — към т.9.
- Край.

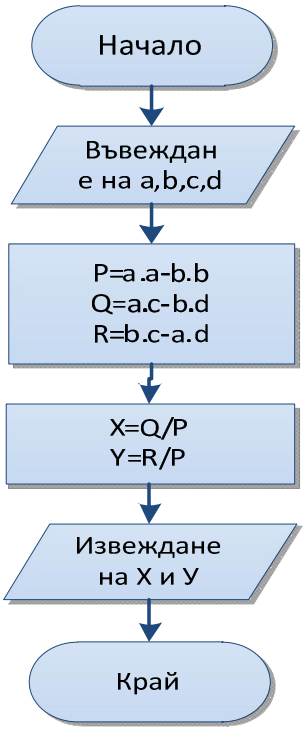
Същият алгоритъм, изразен чрез блокова схема, е представен на фиг. 1.2. Действията, които се извършват, се записват в блок за обработка (функционален блок), а условията които се анализират – в блок за разклонение (логически блок). От структурна гледна точка алгоритмите могат да се разграничат линейни (неразклонени), разклонени и циклични части. На практика всички програми включват еднократно или многократно всички тези части.

Линейната част на алгоритъма е последователност от действия, които се изпълняват винаги в един и същ ред и се представя с последователно свързани блокове с един вход и един изход.

За всяка възможна комбинация от входни данни за съответната част на задачата последователността от действия се запазва и следователно не се налага да се включват условия, които могат да изменят тази последователност.



фиг. 14.



Фиг. 15

Пример. Да се състави алгоритъм за определяне на χ и y от зависимостта $x+iy=(a+ib)/(c+id)$, където $x = (ac - bd)/(c^2 + d^2)$; $y = (ac - ad)/(c^2 + d^2)$; $(c^2 + d^2) \neq 0$

Блоковата схема на алгоритъма е показана на фиг.14. Първият блок

след началния означава въвеждане на входните данни (в случая стойностите на коефициентите (*a,b,c,d)*. Следващият блок е функционален и има предназначението да изчисли междинните величини *P,Q* и *R*. Те ще се изчисляват винаги в посочения ред. По принцип *P.Q* и *R* могат да се изчисляват в произволен ред, което дава възможност те да се обединят в един блок за обработка. За да се изчислят *P,Q* и *R* в друг ред, трябва да се разменят местата на изразите в блока. Във втория функционален блок се определят крайните резултати *X* и *Y*.

Възможно е стойностите на *X* и *Y* да се пресмятат направо. Този вариант е по-неикономичен от гледна точка на броя на действията, тъй като величината *P* се изчислява два пъти.

Необходимостта от проверка на някакво условие (условия), в зависимост от изпълнението на което (които) изчислителният процес протича по един или друг начин, внася разклонение в алгоритъма. В блоковите схеми на алгоритмите разклоненията се реализират с блокове от типа β на фиг.1.1. Най-често се осъществяват разклонения в два клона по следните признаци: положителна или отрицателна стойност, равно или различно от нула, съвпадение или несъвпадение на две стойности и др. Когато се налага разклонение в повече клонове, препоръчва се те да се представят чрез подходящо свързване на блоковете за разклонение в два клона.

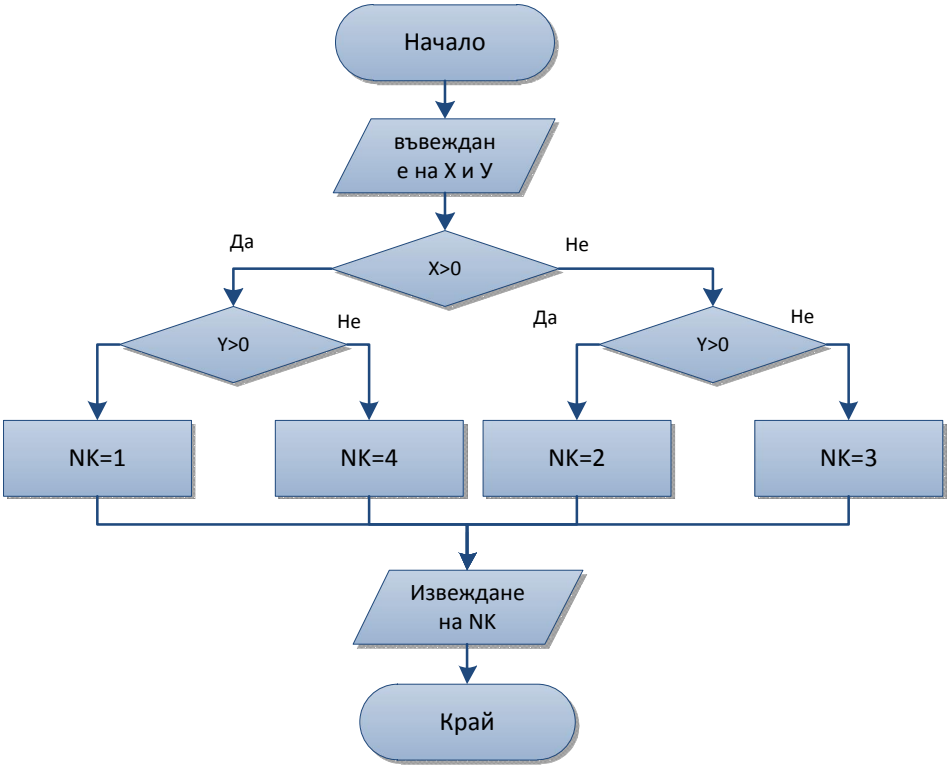
Пример. Да се състави алгоритъм за определянето на квадранта, в който се намира точка, зададена с координатите си A(X,Y).

Точката е зададена с координатите си A(X,Y) и следователно те могат да се въведат като входни данни. За да се определи номерът на квадранта, в който е точката, необходимо е да се проверят *X* и *Y* по отношение на нулата. В дадената на фиг. 15 блокова схема проверката започва с координатата *X*. Възможно е най-напред да бъде проверена координатата *Y*. Проверката на *X* или *Y* свежда възможното множество от четири квадранта на два, след което следващата проверка за *Y* или *X* конкретизира съответния квадрант. В променливата *NK* се записва номерът на квадранта и съдържанието й се извежда като краен резултат.

Пример. Да се състави блокова схема на алгоритъма за определяне на *X* от израза *X = f(Y)*-26.3, където *Y = Z + 2*.

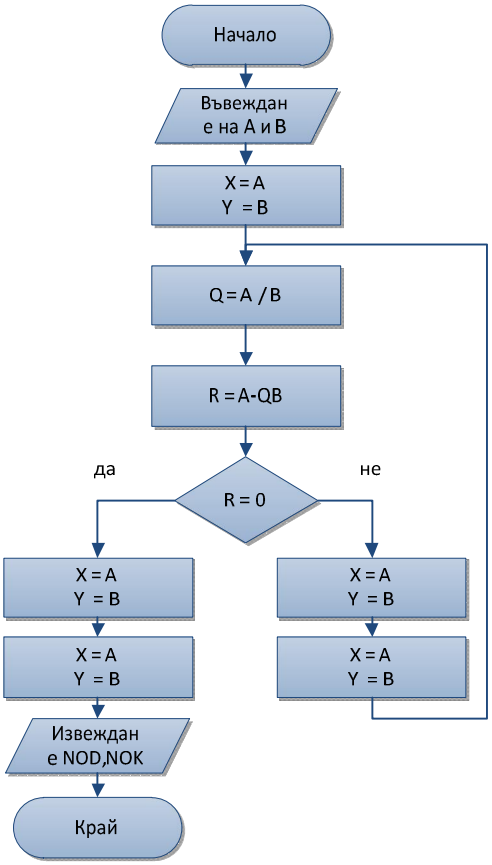
$$f(Y)=\begin{cases} Y^3-0.3, & при \quad Y<0 \\ 0 & при\ Y\le Y\le 1 \\ Y^2+Y & при\ Y>1 \end{cases}$$

За определянето на *X* е необходима стойността на *f(Y)*-която се изчислява в зависимост от Y. Блоковата схема на алгоритъма е представена на фиг. 1.6. Входната величина е Z. Първият функционален блок определя *Y*. В зависимост от конкретната стойност на *Y* изчислителният процес продължава по един от три клона. Изчислената стойност за *f(Y)* се записва в клетка със символичното име F. Независимо от това, по кой от трите клона е определена стойността на *f(Y)*, следващ в блоковата схема е блокът за определяне на *X*. Стойността на *X* се извежда като краен и единствен резултат.

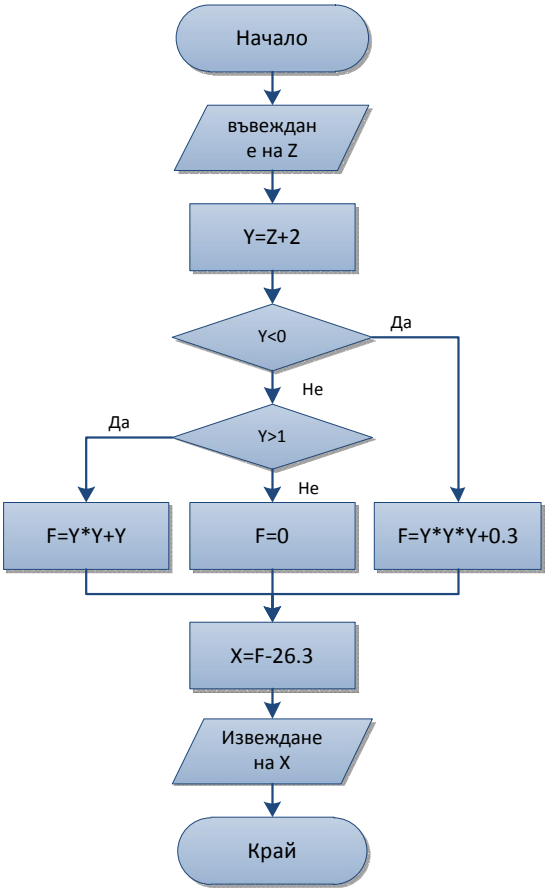


Фиг. 16

Пример: Да се състави блоковата схема на алгоритъма за определяне на най-малкото общо кратно (НОК) на две дадени цели числа *A* и *B* (*A* > *B*).



фиг. 17



Фиг. 18

Методът се свежда до изчисляване на най-големия общ делител (НОД) на числата и определяне на НОК по формулата: *НОК = (АВ) / НОД (А,В)*

Блоковата схема на алгоритъма е показана на фиг.17

За да се определи НОД на двете числа, необходимо е:

1. Разделяне на двете числа. При това се има предвид, че както *А* и *В*, така и частното, което се получава, са от цял тип. Ако *Q* и *R* са съответно частното и остатъкът, *А — BQ + R.*

2. Анализиране на остатъка. Ако остатъкът *R* = 0, НОД = *В* и НОК се определя по посочената формула. Ако остатъкът *R* ≠ 0, необходимо е да продължи делението, като променливите *А* и *В* изменят своето съдържание (на *А* се присвоява стойността на числото *В: : А = В, а в* клетката *В* се записва стойността на остатъка R: *В = R).* Променливите *X и Y* дублират първоначалните стойности на *А* и *В,* с които се оперира при определянето на НОК. Извеждат се два резултата — NOD и NOK. Определянето на НОД е възможно винаги, тъй като 1≤НОД≤В. Процесът се прекъсва най-много след В стъпки;

При обработката на информацията често се налага да се повторят многократно едни и същи действия, като се сменят стойностите на някои параметри. Би могло тези действия да бъдат описвани многократно (всеки път при тяхното срещане) независимо от еднообразието им, но за предпочитане е възможността операциите (действията) да се запишат еднократно. Така се изгражда *цикъл,* който в блоковата схема на алгоритъма представлява затворен път. В цикъла се различават три съставни части: подготовка на цикъла, тяло на цикъла и проверка на условието за край на цикъла (фиг.19а).

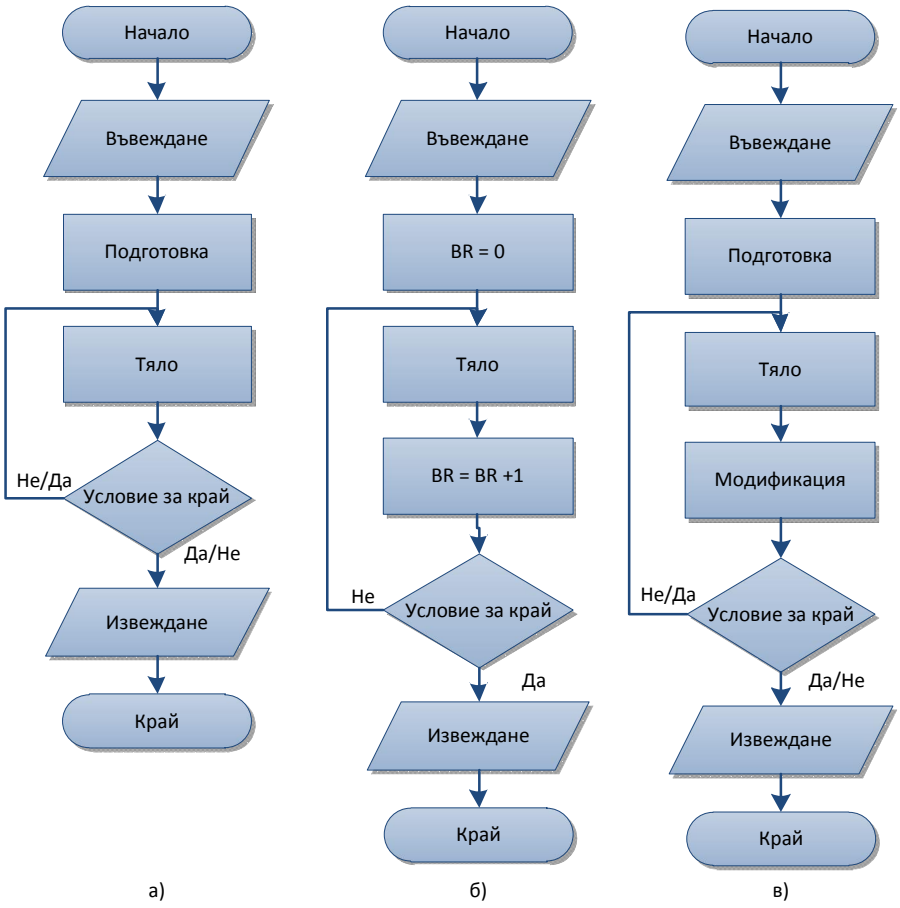
При *подготовката на цикъла* се задават началните стойности на някои величини, които се изменят в процеса на изпълнението на цикъла.

Тялото на цикъла съдържа операцията или съвкупността от операции, които се изпълняват многократно. В него се променят и стойностите на параметрите, като те се подготвят за следващо повторение.

Проверката на условието за край на цикъла осигурява прекратяване на повторението, когато е изпълнено определено логическо условие, наречено условие за край на цикъла.

Условието за край на цикъла (изход от цикъла) може да се зададе

по различни начини. Най-често се срещат следните два начина за определяне края на цикъла:



Фиг. 19

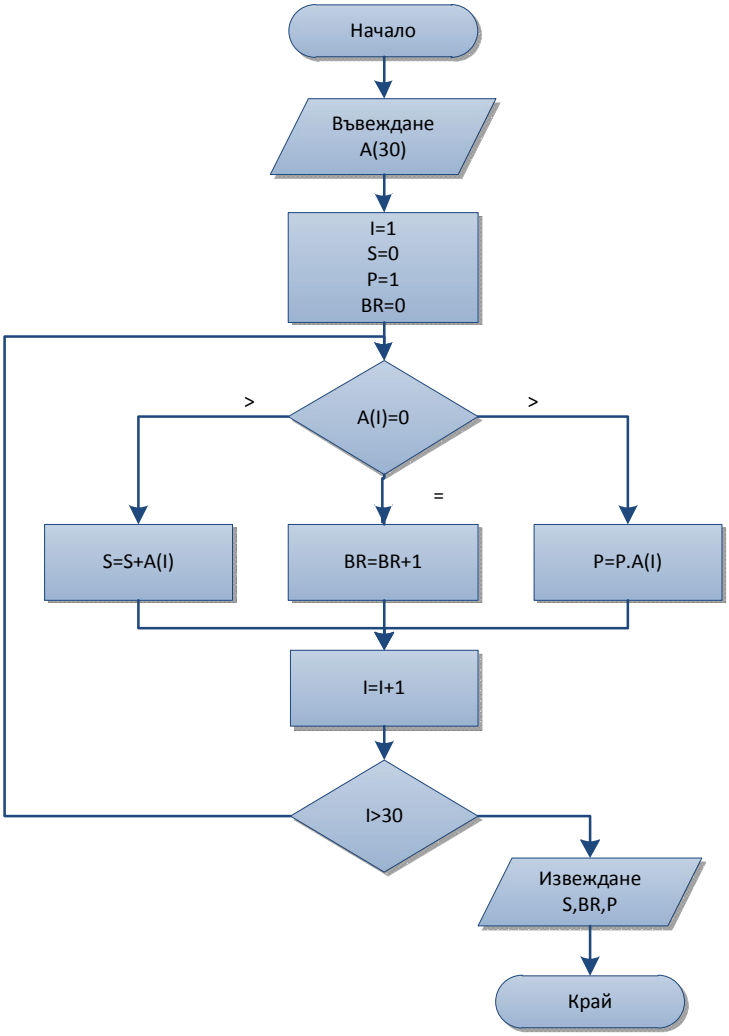
1. Зададен е фиксиран брой повторения на тялото на цикъла. На фиг. 19 б) е дадена обобщена блокова схема на алгоритъм от този тип. В този случай се въвежда т.нар. *брояч на повторенията*. Подготовката на цикъла се състои в присвояване на начална стойност на брояча (в частен случай 0). При всяко изпълнение на тялото се изменя съдържанието на брояча (в частен случай се прибавя 1). От цикъла се излиза, когато съдържанието на брояча достигне определена стойност (в частен случай N)

2. Броят на повторенията на цикъла не е фиксиран предварително.Тялото на цикъла се повтаря, докато обработваната информация удовлетвори определени изисквания, явяващи се условие за край на цикъла (фиг. 19в).

Проверката на условието за край на цикъла може да се извършва преди или след тялото на цикъла (цикли с предусловие и цикли с постусловие).

Много често при съставянето на алгоритмите се използват т.нар. *вложени цикли*. При тях циклите се влагат един в друг. като всеки вътрешен цикъл е част от тялото на съответния му външен. Това обстоятелство определя и основното правело, което трябва да се спазва задължително при работа с вложени цикли: пресичането на *цикли е недопустимо*.

Пример. Да се състави блоковата схема на алгоритъм за определяне на сумата от отрицателните елементи, произведението от положителните и броя на нулевите елементи на дадеш вектор A с 30 елемента.



фиг. 20

По принцип алгоритмът не е свързан с ЕИМ. но с учебна цел тук и по-нататък са използвани някои термини, свързани с машинната му реализация. Елементите на вектора *A* (конкретните 30 числа) трябва да се въведат в 30 последователни клетки на паметта. Всеки пореден елемент се анализира с цел да се определи към коя от трите групи принадлежи (положителни, отрицателни или нулеви), както и да се обработи съобразно с определената група. Следователно в случая е необходимо да се организира цикъл, който ще се повтаря краен (известен) брой пъти, т.е. това е цикъл от първия тип — с брояч (фиг.19б). Броячът на цикъла се установява първоначално в 1 и след всяко изпълнение на цикъла увеличава съдържанието си с 1, докато стигне крайната стойност — 30. По такъв начин съдържанието на клетката, отделена за брояч, определя номера на поредния разглеждан елемент на *A*. В блоковата схема, представена на фиг.1.8, броячът се означава , със символичното име I, а с *A(I)* се определя I-тият елемент, т.е. конкретната стойност, която се съдържа в I-тата клетка на вектора. За трите търсени величини — сума, произведение и брой, се използват три клетки съответно с имена *S*, *P*, *BR*. Съдържанието на тези клетки се изменя в хода на изпълнение на цикъла и следователно последните трябва да бъдат начално установени при подготовката на цикъла (когато се установява и броячът I). Всеки път, когато разглежданият елемент е отрицателен, към съдържанието на *S* се

прибавя неговата стойност, т.е. S трябва да бъде установена първоначално в 0. По аналогични съображения *P* и *BR* трябва да се заредят първоначално съответно с 1 и 0. В клетката *P* се натрупва произведението, т.е. на всяка стъпка съдържанието на клетката *P* се умножава с I-тия елемент, ако той е положителен. Нулевите елементи се броят, като се прибавя 1 към съдържанието на *BR* всеки път, когато се срещне нулев елемент. След като се проверят всички елементи на вектора, т.е. когато I достигне 30, се излиза от цикъла. Извеждат се трите резултата — *S, P* и *BR*

Въпроси и задачи за самостоятелна работа

1. Характеризирайте с конкретни примери основните етапи на алгоритмизацията.
2. Опитайте се да решите приведената комплексна задача по няколко начина и с използването на различни програмно-алгоритмични конструкции.
3. Посочете характерните свойства на алгоритмите.
4. Какво е предназначението на всеки от блоковете за описание на алгоритмите?
5. Представете чрез блокови схеми алгоритмите за решенията на задачите, посочени като описателни примери в текста по-горе.
6. С какво се характеризират логическите алгоритми? Формулирайте правилно и решете самостоятелно една логическа задача.
7. Представете чрез блокова схема алгоритъма за решаване на едно пълно квадратно уравнение с едно неизвестно от вида $a.x^2 + b.x + c = 0$.
8. Опишете алгоритъма за пресмятане на лице на триъгълник със страни *a, b* и *c* (със стойности различни от нула) по формулата на Херон:
   $S = (p(p-a)(p-b)(p-c))^{-1/2}$, където $p = ½.(a + b + c)$.
9. Алгоритмът да включва предварителна проверка на условията за физическо съществуване на триъгълника:
   $a < b + c$; $b < a + c$; $c < a + b$.
10. ***Задача:*** Да се разработи и документира алгоритъм, гарантиращ поддържането на температурата в едно помещение в границите на 18-25 градуса по Целзий.
    - При условие, че температурата спадне под работния интервал, то да се включва нагревател.
    - Ако температурата се покачи над работния интервал, то да се включи вентилатор за охлаждане.
    - При температура под 13 и над 30 градуса – да се задейства звуков и светлинен сигнал за опасност с конкретни съобщения за причината,.
    - Измерването и регулирането да става на всеки 5 минути, а при невъзможност да се реализира някоя от функциите – да се задейства звуков и светлинен сигнал за опасност с конкретни съобщения за причината.

Допълнителна литература

11. http://en.wikipedia.org/wiki/Algorithm
12. http://bg.wikipedia.org/wiki/Алгоритми
13. http://www.programirane.org/