

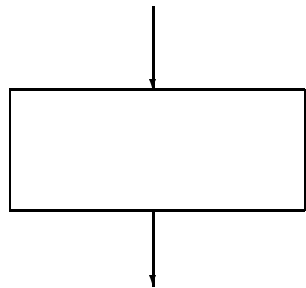
### 3.5 Крайни машини със стекова памет

В този параграф ще дадем още едно описание на безконтекстните езици и ще разгледаме машини, които са еквивалентни на недетерминирани магазинни автомати и са обобщение на тези автомати.

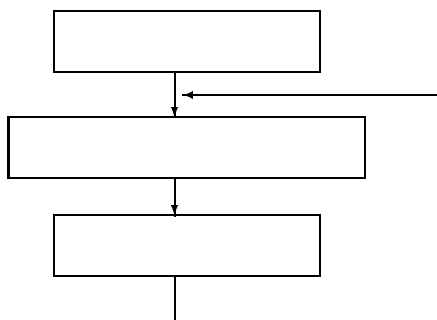
Без да ограничаваме общността, ще разглеждаме езици над азбуката от две букви  $V = \{a, b\}$ . Описанието на крайните машини със стекова памет ще осъществим на езика на *блок-схемите*.

Най-общо в блок-схемите се използват три вида блокове, които се свързват с ориентирани ребра (стрелки). Тези блокове са: *блок за действие (аритметичен блок)*, *блок за анализ (логически блок)* и *служебни блокове*.

Аритметичните блокове се представят графично, чрез правоъгълник с един вход и един изход

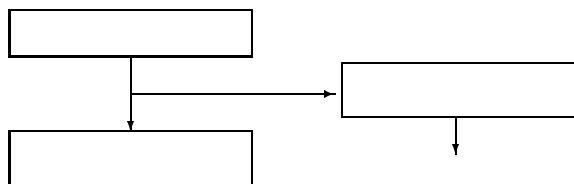


При влизане в този блок винаги се изпълняват действията, предписани в него и се преминава в следващия блок през определения изход. Предписанията за действията в аритметичния блок, обикновено се записват на някои от говоримите езици или пък с използването на специален метаязик (например с математически символи). Това, че даден блок има един вход означава, че при влизане в този блок се извършват едни и същи действия, но този вход може да бъде непосредствено предшествуван от изходи на повече от един блок (Фиг.3.5.1)



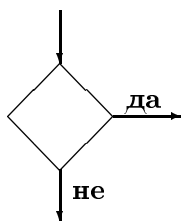
Фиг. 3.5.1

Единствеността на изхода от даден блок означава, че след изпълнението на действията в този блок, се преминава към един единствен, следващ непосредствено блок. Така например недопустимо е използването на блок за действие, както е показано на Фиг.3.5.2



Фиг. 3.5.2

Блокът за анализ се представя, посредством многоъгълник, който има един вход и повече от един изходи

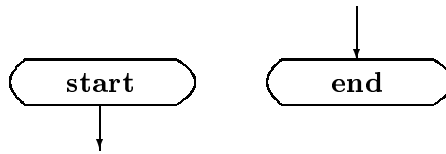


Чрез такъв блок се осъществява алтернативност в работата на машините, в зависимост от данните, с които се работи

в даден момент. В ромба на блока за анализ се записва някое логическо условие (съждение), верността на което се проверява и в зависимост от това се прави избор по кой от изходите ще продължи работата на машината. Често блоковете за анализ се наричат логически блокове или блокове за условен преход.

Обикновено, ако логическото условие е удовлетворено, изпълнението продължава към изхода, обозначен с "да", а в противен случай към изхода, обозначен с "не".

Служебните блокове се използват за означаване на началото и края на блок - схемата. Обикновено, те са във вид на овали и в тях се записва предназначението (съдържанието) им



Всеки блок за начало има само един изход и нито един вход, докато блоковете за край имат единствен вход без нито един изход.

Често блоковете в дадена блок-схема се наричат *оператори*.

Най-важното приложение на блок-схемите е в описанието на различни алгоритми. В самите алгоритми, заради тяхната гъвкавост, по-голямата част от обектите, с които се работи се означават с букви, които ще наричаме *променливи*. При описанието на алгоритмите, чрез блок-схеми техните променливи се явяват, като променливи за самите блок-схеми. По-нататък ние ще разглеждаме блок-схеми, в които променливите приемат за стойности думи над азбуката  $V = \{a, b\}$ , ако изрично не е посочена някоя друга азбука.

Нека  $x$  е променлива, чиито стойности са във  $V^*$  и да разглеждаме следните функции на  $x$ :

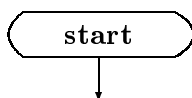
$$\text{first}x = \begin{cases} u & \text{ако } x = u\alpha, \ u \in V \\ \varepsilon & \text{ако } x = \varepsilon, \end{cases}$$

$$\mathbf{tail}x = \begin{cases} \alpha & \text{ако } x = u\alpha, u \in V \\ \varepsilon & \text{ако } x = \varepsilon. \end{cases}$$

т.е. функцията **first** $x$  дава първата буква на думата  $x$ , а **tail** $x$  задава думата, получена от  $x$ , след като се отстрани от нея първата ѝ буква.

**Определение 3.5.1** Под **крайна машина**  $M$  над азбуката  $V$  ще разбираме блок-схема с една променлива  $x$ , приемаща стойности в  $V^*$ , в която операторите са от следните видове:

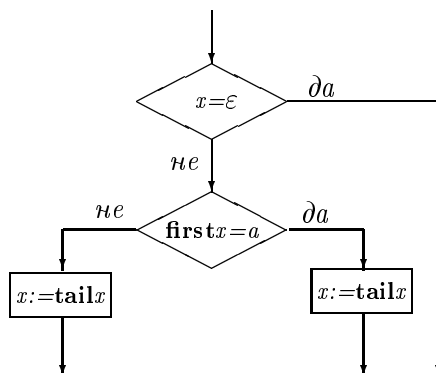
1. Оператор за начало **start**



2. Оператори за край

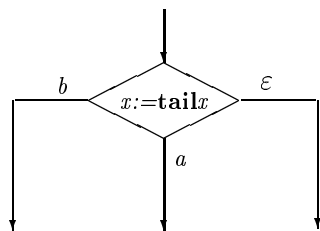


3. Оператор за анализ



Фиг. 3.5.3

който за по-кратко ще означаваме по следния начин

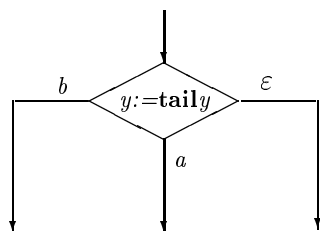


Фиг. 3.5.4

Езикът на блок-схемите ни дава възможност за една нова дефиниция на понятието стек.

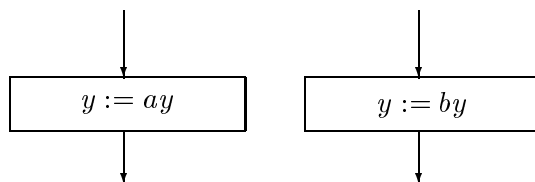
**Определение 3.5.2** *Една променлива  $y$  над дадена азбука  $V'$  се нарича **стек** (стекова памет, стекова променлива), ако върху нея могат да се приложат само някои от следните оператори:*

1. Оператор за проверка



Фиг. 3.5.5

2. Оператори за присвояване



Фиг. 3.5.6

Ясно е, че в стековата памет може да се записва или чете само буквата **firsty**, т.е. както вече казахме при обработването на стекове е в сила принципа ”последно въведено – първо изведено”.

Крайна машина с една променлива  $x$  може да работи без стекова променлива, с една или с повече стекови променливи.

Работата на крайната машина се състои в това, че на входа на машината се задава стойност на променливата  $x$  и прилагайки върху нея, последователно зададените оператори на машината се достига до някои от операторите за край **admit** или **reject** или пък не се достига до такъв оператор. В последния случай се казва, че машината *зацикля* когато работи върху думата  $x$ .

Да отбележим, че когато машината зацикля при работата си с думата  $x$ , възниква проблем, с установяването на този факт (че машината зацикля). Наистина, за това не може да се съди само от работата на машината, тъй като самият процес на зацикляне е вечен.

Често доказването, че една крайна машина не завършва работата си (зацикля) с входната дума  $x$ , има характер на математическо доказателство.

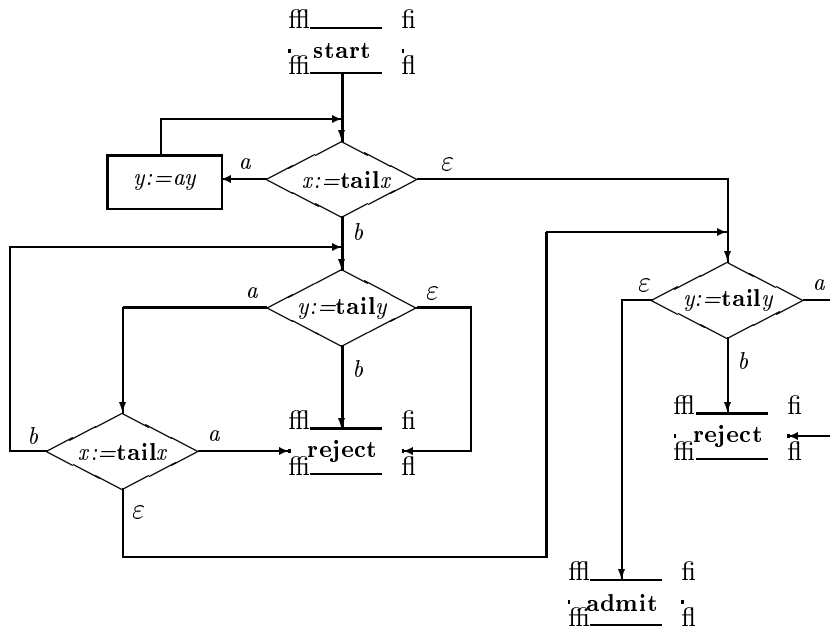
**Определение 3.5.3** *Казваме, че една машина **разпознава** (отхвърля) думата  $\omega \in V^*$ , ако започвайки работа върху  $\omega$  достига до оператора **admit** (**reject**).*

Да означим с  $L(M)$  множеството от всички думи  $\omega \in V^*$ , за които машината  $M$  завършва своята работа в оператор *admit*. Ще казваме още, че езикът  $L(M)$  се *разпознава* (*поражда*) от машината  $M$ .

В този параграф ще разгледаме крайни машини с един и с два стека.

Може да се докаже, че крайните машини с един стек са еквивалентни (пораждат едни и същи езици) с детерминирани магазинни автомати.

**Пример 3.5.1** Да разгледаме машината  $M_1$ , която поражда езика  $L = \{a^n b^n \mid n \geq 1\}$ , който както вече показахме е безконтекстен, но не е автоматен. Тази машина е представена на Фиг.3.5.7



**ФИГ. 3.5.7**

Крайна машина с един стек, която разпознава езика  $\{a^n b^n | n \geq 1\}$ .

Да проследим работата на машината при входна дума  $w = aaabbb$ . В началното положение, стекът  $y$  има стойност празната дума  $\epsilon$ .

След като прочете първите три букви на  $x = \omega$ , машината изпълнява трикратно оператора  $y := ay$ . Следователно стойностите на променливите ще бъдат  $x = bbb$  и  $y = aaa$ .

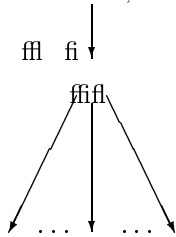
След четвъртата от началото на думата  $\omega$ , входна буква  $b$ , състоянието на променливите се изменя, като за всяко извадено  $a$  от  $y$  се изважда **first** $x$  и ако **first** $x = b$ , то от  $y$  се изважда **first** $y$ . Следователно променливите  $y$  и  $x$  приемат последователно стойностите  $x = bb, y = aa, x = b, y = a, x = \varepsilon, y = \varepsilon$  и в този

момент се извършва проверка дали  $y = \varepsilon$ , което е достатъчно за да бъде думата  $\omega$  разпозната от дадената машина  $M_1$ .

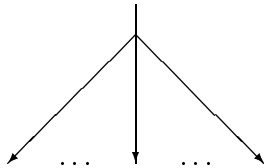
След този пример, може да резюмираме, че в първия цикъл, машината изважда от променливата  $x$  максималния ѝ префикс  $\alpha$ , който се състои само от  $a$ -та и същият се зарежда в стека  $y$ . В следващия етап от работата си машината изважда от  $x$  и  $y$  последователно буквите им, като следи за всяко извадено  $b$  от  $x$  да се изважда по едно  $a$  от  $y$ . Ако това се наруши, машината отхвърля думата  $\omega$ . Също така, когато в  $x$  са извадени всички букви, същото трябва да е станало и с тези в  $y$  т.е. стекът трябва да е празен, в противен случай  $\omega$  отново се отхвърля.

**Определение 3.5.4** *Една крайна машина ще наричаме **недетерминирана**, ако в нейната блок-схема се допускат преходи по избор т.е. допускат се разклонения, които не зависят от някакво логическо условие, а е налице възможност за безусловен избор за продължаване работа на машината.*

Това означава, че може да се използва оператор от вида



или за краткост можем да смятаме, че стрелките в блок схемата могат да имат следното представяне



При работа на машината, когато достигнем до такава точка от блок-схемата ѝ можем да изберем произволен изход и да



продължим по-нататък. Ясно е, че при работата си такава машина върху дадена дума, може да има повече от едно изпълнение, в зависимост от това, кое продължение сме избрали във всяка точка на "безусловно разклонение" на блок-схемата.

**Определение 3.5.5** *Ще казваме, че думата  $\omega$  се разпознава от недетерминираната крайна машина  $M$ , ако съществува изпълнение на  $M$  върху думата  $\omega$ , което завършва с оператора **admit**.*

**Пример 3.5.2** Да се построи крайна машина, която разпознава езика  $L = \{\omega \mid \omega = \omega_1\omega'_1, \text{ където } \omega'_1 \text{ е дума получена, като запишем } \omega_1 \text{ в обратен ред}\}$ .

Да забележим, че този език е безконтекстен. Наистина, той се поражда от граматика със следните правила

$$S \rightarrow aSa|bSb|aa|bb.$$

В езика  $L$  се включват например думите  $\varepsilon, abba, aabbbaa, bbabbabb$  и т.н. Това са думи, които са с дължина четно число, като спрямо средата на думата буквите ѝ имат огледално разположение. От тук е ясно, че за да се разпознават такива думи от дадена крайна машина  $M$ , тя трябва да може да намира средата на всяка дума и след това да извършва проверката за огледалност на думата спрямо нея. Ако е дадена думата  $\omega = a_1a_2\dots a_s \in V^*$ , то намирането на средата на тази дума може да се реализира, например по един от следните начина:

(i.) Извеждат се по двойки буквите от  $\omega$  и за всяка такава двойка в стека  $y$  се зарежда по една буква. Ако след последната изведена двойка от  $\omega$  са изчерпани всички букви на  $\omega$ , то дължината на стека  $y$  ни задава средата на думата  $\omega$ , а в противен случай думата  $\omega$  има за дължина нечетно число и следователно не е в  $L$ .

(ii.) Вторият начин се състои в следното: за всяко  $i$ ,  $1 \leq i \leq s$  проверяваме дали  $a_i$  е средата на  $\omega$  и тази проверка се провежда заедно с решаването на цялата задача. При избрано  $i$  това става, като отведем думата  $\omega_1 = a_1a_2\dots a_i$  в стека  $y$  и след това проверяваме дали думата  $\omega_2 = a_{i+1}a_{i+2}\dots a_s$  е равна на

думата  $\omega'_1 = a_i a_{i-1} \dots a_1$ . Да отбележим, че последната проверка се осъществява, като последователно се сравняват буквите  $a_i$  с  $a_{i+1}$ ,  $a_{i-1}$  с  $a_{i+2}$  и т. н.

Да забележим, че и при двата начина машината трябва да използва стекове. Нещо повече при първия начин в стека  $u$  се съдържа информация само за това къде е средата на дадената дума  $\omega$ . Ето защо е необходимо да използваме поне още един стек, в който да отведем първата половина на думата  $\omega$ , която ще сравняваме с останалата част на  $\omega$ .

При вторият начин задачата се решава, като "изпробваме" дали  $a_i$  за някое  $i \leq s$ , е среда на думата  $\omega$ . Следователно имаме възможност да избираме: да правим проверката дали  $a_{i+1} a_{i+2} \dots a_s = (a_1 a_2 \dots a_i)'$  или да увеличим  $i$  с 1, докато  $i$  стане равно на  $s$ . Следователно в този случай можем да опишем езика  $L$  с недетерминирана крайна машина  $M_3$  с един стек, докато във случая, описан в (i), същият език може да се опише с крайна машина  $M_2$  с два стека. На Фиг.3.5.8 и Фиг.3.5.9 са дадени блок-схемите на машините  $M_2$  и  $M_3$ .

От направените по горе разсъждения се вижда, че езикът  $L = L(M_2) = L(M_3)$  не може да бъде поражен от крайна детерминирана машина с по-малко от два стека, което означава, че  $L$  не е детерминиран език. Също така с този пример се вижда, че недетерминираната машина с един стек  $M_3$  няма еквивалентна крайна детерминирана машина с един стек.

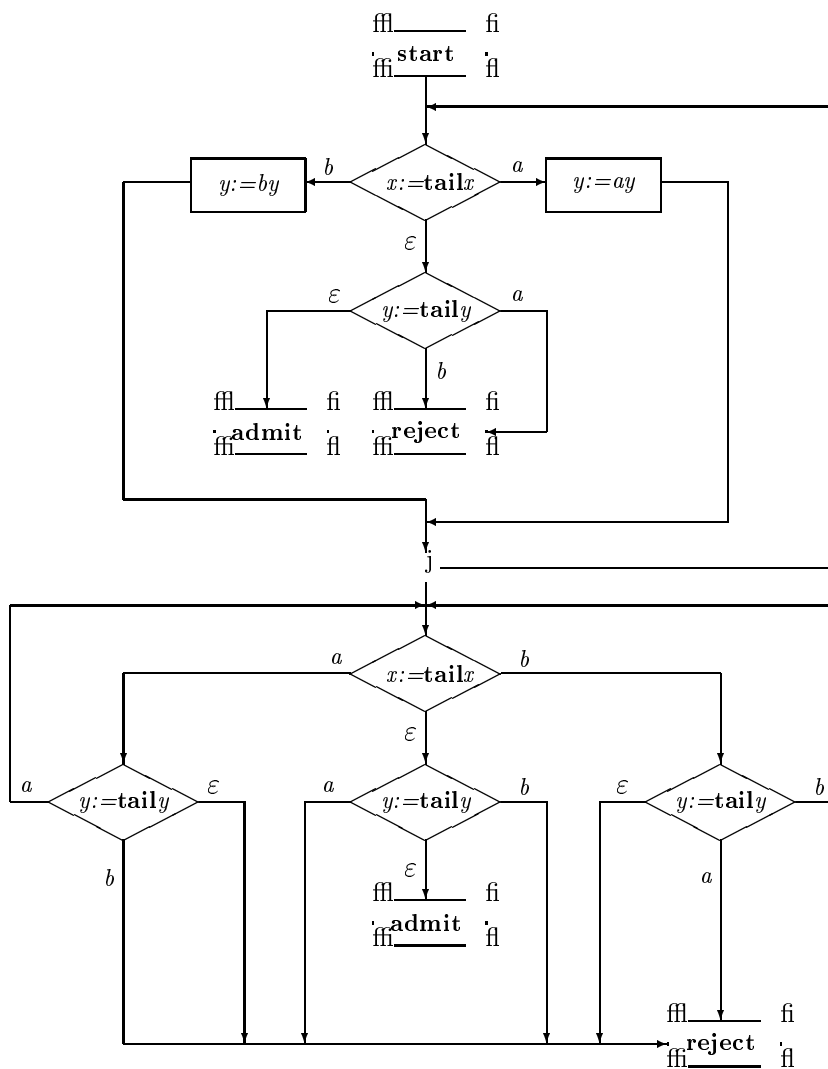
От друга страна съществува машина с два стека, която няма еквивалентна крайна машина с един стек. Това е същината на следващото упражнение.

**Пример 3.5.3** Да се построи крайна машина  $M_4$  с два стека, която поражда езика  $L = \{a^n b^n a^n \mid n \geq 1\}$ .

Преди да опишем как ще работи  $M_4$  да забележим, че от Теорема 3.4.3 следва, че  $L$  не е безконтекстен език и следователно, крайна машина с един стек не може да породява  $L$ .

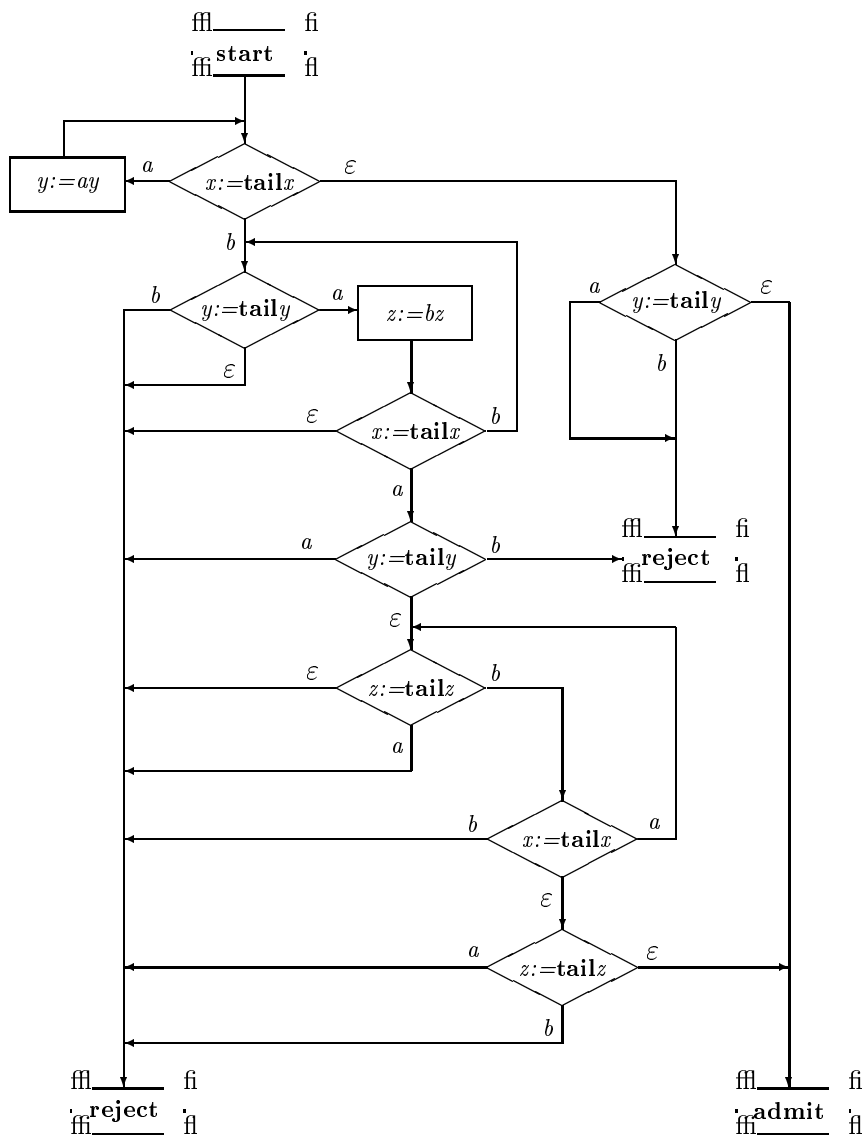
Да опишем как трябва да се построи машината  $M_4$  т.е. как би работила тя върху произволна входна дума  $\omega = a_1 a_2 \dots a_s$ .





Фиг. 3.5.9

Недетерминирана машина с един стек, която разпознава езика  $\{\omega\omega' \mid \omega \in \{a, b\}^*\}$ .



Фиг. 3.5.10

Крайна машина с два стека, която разпознава езика  $\{a^n b^n a^n \mid n \geq 0\}$ .

Един възможен алгоритъм за работа на машината  $M_4$  се състои в следното: най-напред се преместват в стека  $y$  всички  $a$ -та от първата позиция на  $\omega$  до появата на буквата  $b$ . След това се преместват в стека  $z$  всички  $b$ -та от първата позиция на това, което е останало от  $\omega$  до първата поява на буквата  $a$ , като едновременно се следи, след края на тази операция, буквите в  $z$  да са толкова колкото са били тези в  $y$ . Накрая се проверява остатъкът от  $\omega$  да се състои само от  $a$ -та, които са толкова колкото са били  $b$ -та в  $z$ . (Фиг.3.5.10)

### З а д а ч и

1. Да се построи крайна машина, която разпознава езика  $L$ , ако:
  - а)  $L = \{a^j b^j a^i \mid i, j \geq 1\}$ ;
  - б)  $L = \{a^i b^k a^k \mid i, k \geq 1\}$ ;
  - в)  $L = \{a^k b^k \mid k \geq 1\}$ .
2. Да се докаже, че езикът  $L = \{a^k b a^k \mid k \geq 0\}$  не е автоматен.

**У п ъ т в а н е:** Постройте крайна машина, която разпознава езика  $L$ .

3. Да се построи крайна машина, която разпознава езика  $L = \{w \in \{a, b\}^* \mid \text{в } w \text{ има два пъти повече } a\text{-та отколкото } b\text{-та}\}$ . Да се докаже, че не съществува крайна машина без стек, която да разпознава езика  $L$ .