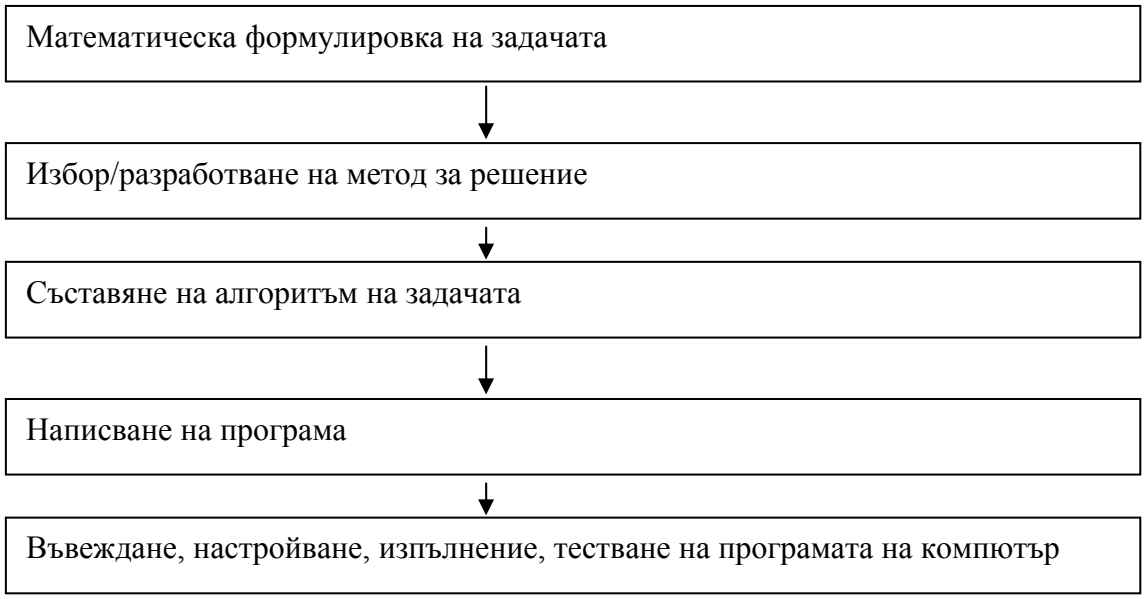
Тема 4
Алгоритми - същност и свойства, начини за описание. Блок-схеми. Базови алгоритмични структури

1. Основни етапи при подготовката и решаването на задача с компютър

Подготовката и решаването на задача с компютър преминава през следните основни етапи:



Етапите, които предхождат програмирането са математическата формулировка, избор на метод на решение и съставяне на алгоритъм. При решаването на конкретна задача някои от тези етапи се изключват от самата постановка на задачата. Например, ако се изисква да се изчисли стойността на функцията $y = ax + в$ за различни стойности на аргумента *x,* то задаването на формулата на тази функция се явява математическа формулировка на задачата и определя метода на нейното решаване.

В общия случай, решаването на задачата ще трябва да премине през всички указани по-горе етапи.

Пример*:* Материално тяло е изхвърлено вертикално нагоре с начална скорост *v.* Да се определи след колко време тялото ще достигне най-високата точка в траекторията си и на какво разстояние от земята ще бъде тази точка.

Математическа формулировка на задачата (формализация)

На този етап условието на задачата се описва или във вид на уравнения, или във вид на последователност от формули, които водят до нейното решение.

За задачата от примера се използват познати от физиката формули. Разстоянието y, изминато от тялото, може да се определи от уравнението:

$y = vt\text{-}gt^2/2$

където *v [m/s]* е началната скорост, g=9.81 [m/s^2] е земното ускорение (постоянна величина) и *t [s]* е времето на движение на тялото.

След математическа преработване на формулите (диференциране по *t*) задачата се описва с две формули:

$t=v/g$

$s=vt\text{-}gt^2/2$

В резултат на формализацията се получава математическият модел на задачата.

Избор или разработване на метод на решение

Този етап не може да се разглежда самостоятелно. Той е свързан с етап формализация, тъй като целта на формализацията е изграждането на математически модел на задачата, за който е известен метод на решение.

За разгледания пример формулите от математическия модел определят и метода на решение.

Това е точен метод на решение, тъй като чрез него се получава точното решение на задачата.

При много класове от задачи (например решаването на алгебрични, диференциални уравнения и системи от уравнения, изчисляване на интеграли и др.) точни методи за решение или не са известни, или се свеждат до тежки и обемни формули. За решаването на основните класове от такива задачи са разработени числени методи за решение. Те са предмет на изчислителната математика. Числените методи дават решението на задачата с предварително зададена точност - т.е. това са методи за приближено изчисляване на резултата. На практика съществуват и могат да се разработят числени методи за всички математически задачи, срещащи се в инженерно-техническата, икономическата и научната дейност, като в същото време за някои задачи могат да бъдат използвани различни числени методи. В такива случаи трябва да се избере най-подходящият метод.

Най-често критерий за избор на метод на решение е ефективността на програмата, която ще бъде разработена по този метод. При това ефективността се избира по следните критерии: време за решаване на задачата с компютър, обем оперативна памет, необходим за нейното решаване, точност на резултата. Най-често при избора на метод на решение се налага да се прави компромис между ефективност и точност.

В случаите, когато не е известен метод за решаването на определена задача или съществуващите методи не удовлетворяват достатъчно изискванията за задоволителна точност, съчетана с добра ефективност, се налага разработването на нови методи за решаване на задачата.

2. Съставяне на алгоритъм на задачата

Понятието алгоритъм е едно от основните в математиката и изчислителната техника. На него се базира използването на компютъра за решаване на различни задачи.

Алгоритъмът е подредено крайно множество от правила, които определят процеса на обработка на информацията. И до днес се използват правилата (алгоритмите) за изпълнение на четирите основни аритметични действия над многоцифрени числа, които са разработени през IX век от узбекския математик Ал-Хорезми. От името на този математик произлиза и терминът алгоритъм.

Алгоритъмът трябва да е така разработен, че дори човек който няма познания в съответната област да разбира и точно да изпълнява съдържащите се указания и да получи решение на задачата.

Алгоритмите трябва да се описват чрез точно определени елементарни действия. Елементарното действия е това, което може да се изпълни без допълнително пояснение. Всяко изпълнение на елементарно действие се нарича стъпка.

Могат да се дадат различни определения за алгоритъм.

Алгоритъм - това е точно и понятно описание на последователност от действия, които се извършват върху входните данни, които водят изпълнителя след краен брой стъпки до постигане на определена цел или решаване на поставена задача.

Друго определение за алгоритъм е това на руският учен А. А. Марков, който е един от основоположниците на съвременната теория на алгоритмите: Точно и общоразбираемо предписание, определящо изпълнението на последователност от елементарни операции,

чрез които се решава клас от еднотипни задачи.

Алгоритъмът може да се изпълнява не само от човек и за тази цел той трябва да се опише по подходящ начин. При изпълнение на алгоритъм с машина е необходимо представянето му в явна форма, придружена с детайлно описание , т.е. показването и на тези стъпки, които биха били изпълнени машинално от човека.

Не всяко описание от действия е алгоритъм.

3. Свойства на алгоритмите

Независимо от това кой е изпълнителят на даден алгоритъм, какви са входните данни и каква е поставената цел, общовалидни са следните свойства на алгоритмите:

1. Дискретност.

Алгоритъмът трябва да се състои от краен брой завършени действия (стъпки). Преходът към всяко следващо действие е възможен само след завършване на предишното.

2. Определеност (детерминираност).

Всеки алгоритъм се построява, за да се разчита от определен изпълнител, затова всяка команда трябва да бъде еднозначно разбрана от изпълнителя

3. Масовост.

С помощта на определен алгоритъм трябва да може да се реши не една конкретна задача, а множество от еднотипни задачи и то многократно.

4. Резултативност.

Изпълнението на алгоритъма винаги трябва да води до получаване на резултат, т.е. до решаване на задачата. За "решаване на задачата" се разбира също и извеждане на съобщение, че при зададените входни данни задачата е нерешима.

4. Описание на алгоритми

Словесно описание на алгоритъм - с думи се описва последователността от действия, които водят до решаването на задачата.

Алгоритъмът на задачата от примера може да бъде описан словесно по следния начин:

1. Да се въведе в оперативната памет на компютъ (например от клавиатурата) стойност за v
2. Да се изчисли стойността на t по формулата $t=v/g$
3. Да се изчисли стойността на s по формулата $s=vt-gt^2/2$
4. Да се изведат (например на монитор) изчисленистойности за t и s
5. Край.

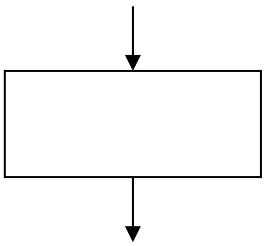
Очевидно, словесното описание на алгоритмите може бъде използвано само за алгоритми с много проста логическа структура.

При описването на алгоритми с по-голяма сложност се използва или описание чрез проектантстки езици, или графично описание.

Графично описание на алгоритъм (описание чрез блок-схема)

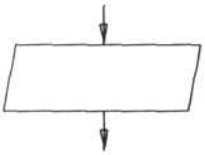
При този начин на описание изпълнението на алгоритъма се представя като движение по геометрична фигура, състояща се от възли (върхове) и свързващите ги клонове. Възлите се наричат още блокове и чрез тях се описват действията, предписани от алгоритъма. Клоновете (свързващите стрелки) указват последователността на изпълнение на действията.

Най-често използваните блокове са:

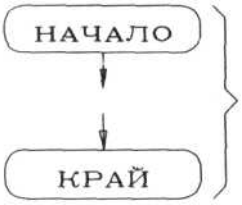


**Блок за обработка* - има един вход и един изход. Вътре в него с думи или с математически символи се описва действието (действията), което трябва да с извърши;

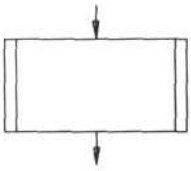
**Логически блок (блок за разклонение)* - има един вход и два изхода. В него изчислителният процес се разклонява в зависимост от някакво логическо условие (изпълнено или неизпълнено);



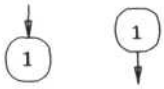
**Блок за въвеждане или извеждане* - означава действията за въвеждане на входни данни към оперативната памет на компютъра и извеждане на резултати от оперативната памет на компютъра към изходните устройства;



**Блокове за начало и край* - поставятсе в началото и края на блоковата схемана алгоритъма;

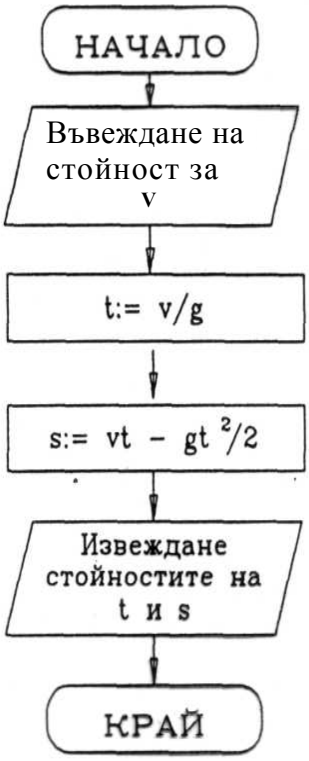


**Блок за извикване на подпрограма* -използва се при записването на сложни алгоритми, при които част от действията са организирани като самостоятелна програмна част (подпрограма);



**Между страничен съединител* - използва се тогава, когато при описаниетона алгоритъма се налага преминаването към нова страница.

Графичното описание на задачата от примера е следното:



Символът ": = " се тълкува като "приема стойността на", т.е. *t* приема стойността на израза *v/g.*

Описание на алгоритми на проектантски език:

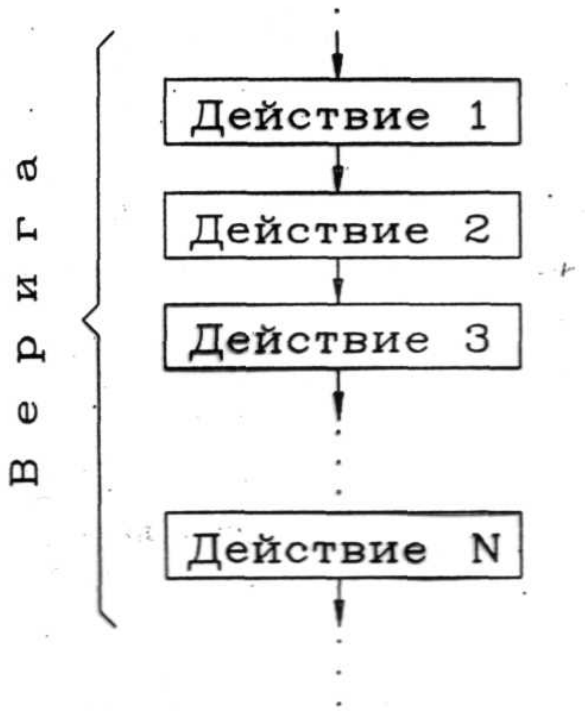
Тези езици заемат междинно място между естествените езици и езиците за програмиране - на естествен език програмистът описва последователността от действия в алгоритъма, но спазвайки правилата на езиците за програмиране. Елементи на проектантски език се използват в словесното описание на базовите алгоритмични структури.

5. Базови алгоритмични структури

Всеки алгоритъм се синтезира (съставя) от прости, относително самостоятелни структурни единици, наречени базови алгоритмични структури. Те са три вида - верига, разклонение и цикъл.

Верига (линеен участък)

Това е "линейна" последователност от действия, т.е. действията, описани в структурата, се изпълняват едно след друго по реда на тяхното подреждане. Такъв ред на изпълнение на действията на алгоритъма се нарича "естествен ред". В частност, веригата може да съдържа само едно действие или да не съдържа действия (да бъде "празна").



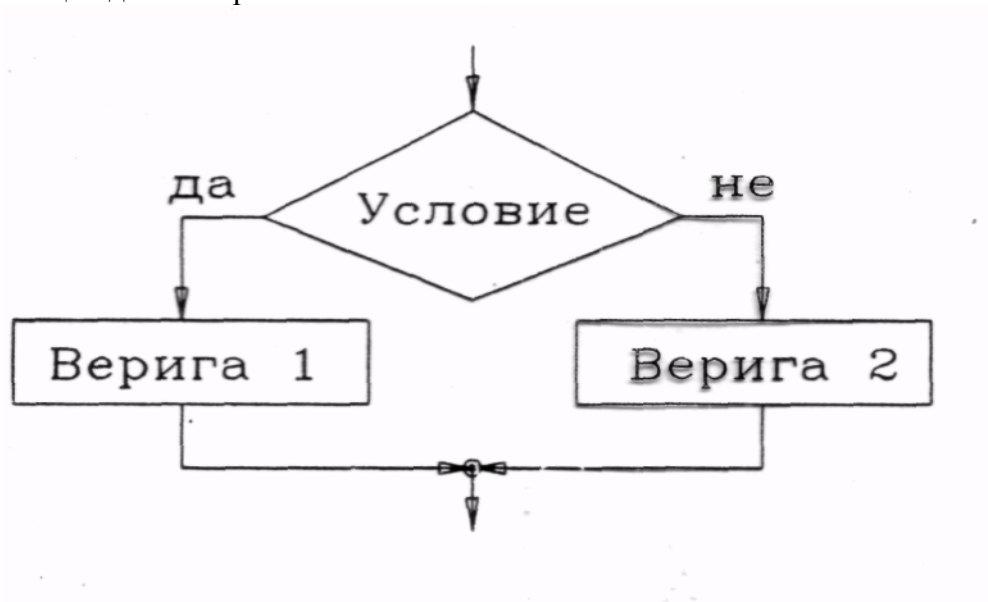
Алгоритми, съдържащи само верига, се наричат линейни алгоритми. Пример за линеен алгоритъм е алгоритъмът на задачата за вертикалното движение на тяло. В практиката на програмирането такива задачи се срещат рядко.

Разклонение

Това е алгоритмична структура, в която изчислителният процес се разклонява в две или повече посоки в зависимост от дадено логическо условие (изпълнено или неизпълнено).

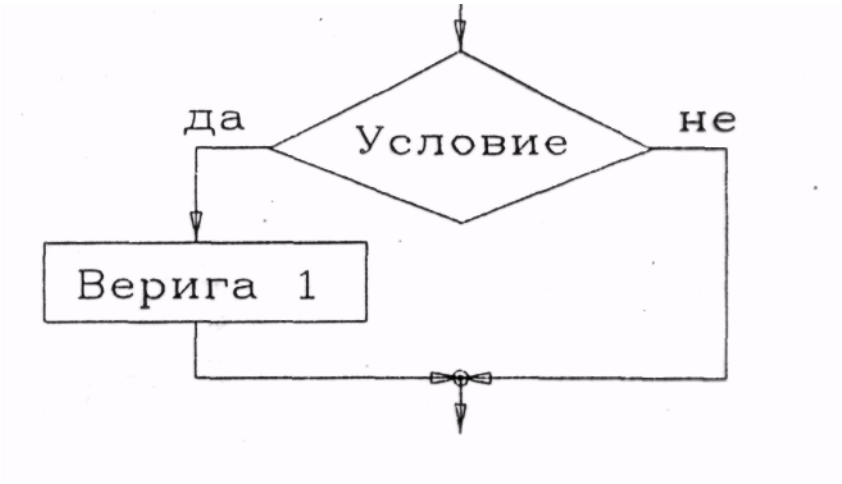
Когато разклонението е двупосочно, то се нарича алтернатива.

Общ вид на алтернативата:



Частен случай на алтернативата е т.нар. алтернатива - кратка форма. При нея веригата в един от двата й клона (но не и в двата едновременно) е празна.

Общ вид на алтернатива - кратка форма

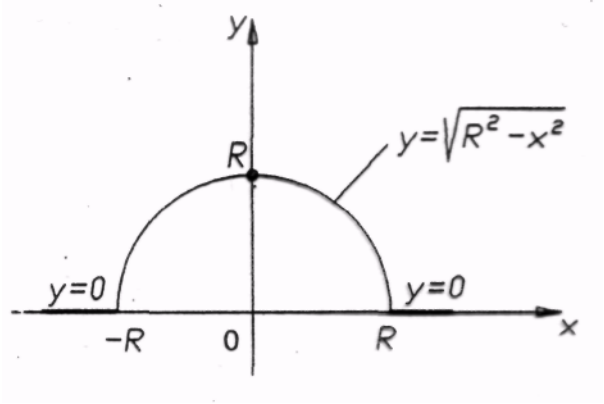


Ако в разклонението изчислителният процес се разклонява в повече от две посоки - говорим за многостранно (многозначно) разклонение и такива случаи се налага в структурата на една алтернатива да се включи друга алтернатива.

Пример за алгоритъм, съдържащ алтернатива:

Да се изчисли, за различни стойности на *x* стойността на функцията *y=f(x),* която е зададена по следния начин: :

$$y = \begin{cases} 0, & \text{за } |x| > R \\ \sqrt{R^2 - x^2}, & \text{за } |x| =< R \end{cases}$$



Съставяне на алгоритъм

Цикъл

Цикълът е част от алгоритъма, която записана само веднъж, се изпълнява многократно за различни входни данни. Частта, обхванатата от цикъл, може да съдържа вериги, разклонения и други цикли. В последния случай се говори за вложени цикли.

Каквито и базови алгоритмични структури да съдържа структурата цикъл, тя винаги се изгражда от 4 основни съставни части:

-инициализация (подготовка за влизане в тялото на цикъла). Задава се начална стойност на величината, която управлява повторението на цикличната част - нарича се управляваща променлива на цикъла;

-тяло на цикъла - съдържа действията (или действието), които ще се изпълняват многократно;

-актуализация - подготовка за следващото изпълнение на тялото на цикъла;

-проверка на условието за край на цикъла - осигурява изход от цикъла (прекратяване на изпълнението на цикличната част).

Общ вид на структурата цикъл

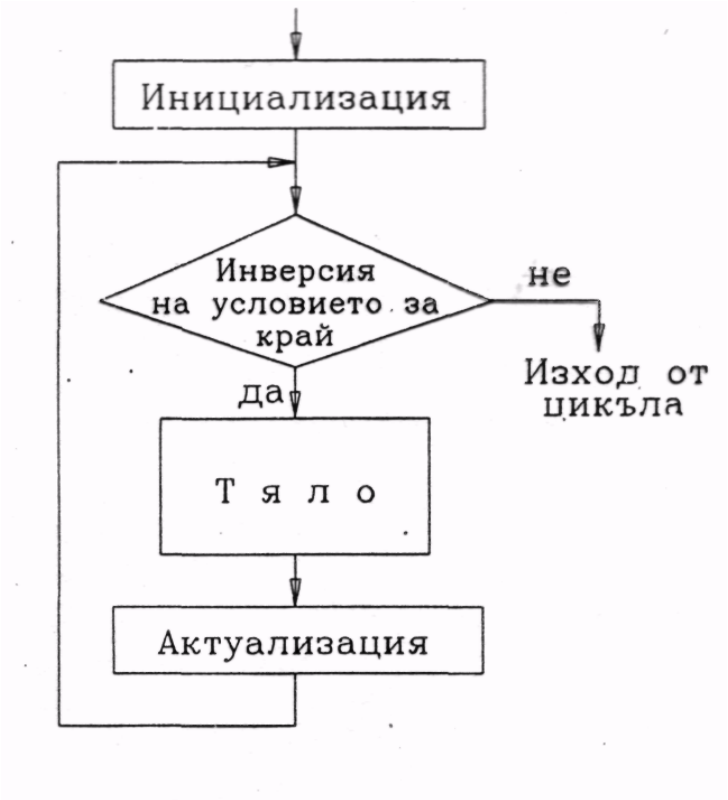
Четирите основни съставни части на цикъла (инициализация, тяло, актуализация и условие за край) могат да бъдат разположени по различен начин, което води до няколко структурни разновидности на циклите.

Видове цикли според структурата

Цикъл с постусловие - проверката на условието за край се прави всеки път след изпълнението на тялото на цикъла. Тялото на цикъла се изпълнява дотогава, докато се изпълни условието за край. Показаната по-горе структура на цикъл е от този вид. Този цикъл се изпълнява поне веднъж.

Цикъл с предусловие - проверката на условието за край (по-точно инверсията (отрицанието) на условието за край, което бихме нарекли "условие за продължаване") се прави всеки път преди изпълнението на тялото на цикъла. Цикълът се изпълнява дотогава, докато е неизпълнено условието за край (изпълнено е "условието за продължаване"). При някои случаи този цикъл може изобщо да не се изпълни.

Общ вид на цикъл с предусловие:



За онагледяване на трите структурни разновидности на циклите може да се разгледа следния пример:

Да се изчисли сумата на първите *N* естествени числа.

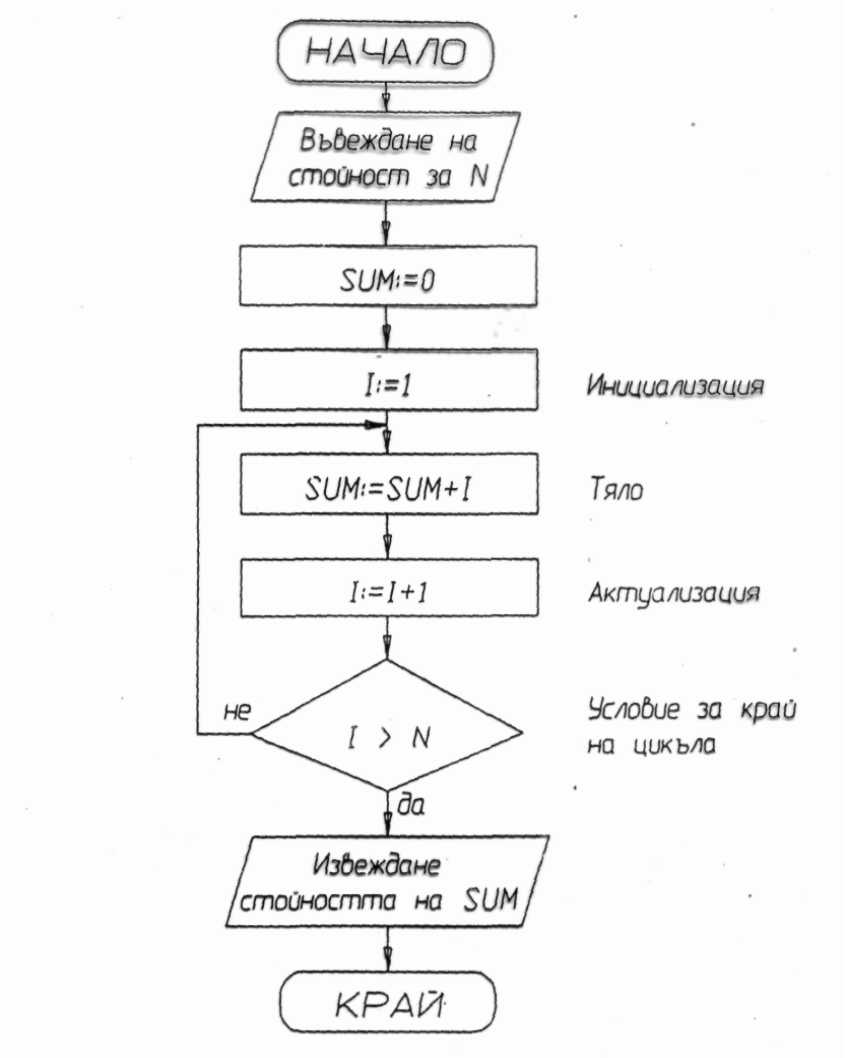
Задачата се свежда до изброяването на целите числа от 1 до N и прибавянето им към текущо изчислената сума (т.е. извършване *N* на брой пъти на действие сумиране).

Въвеждат се следните помощни величини (променливи). На тях съответстват клетки в оперативната памет на компютъра.

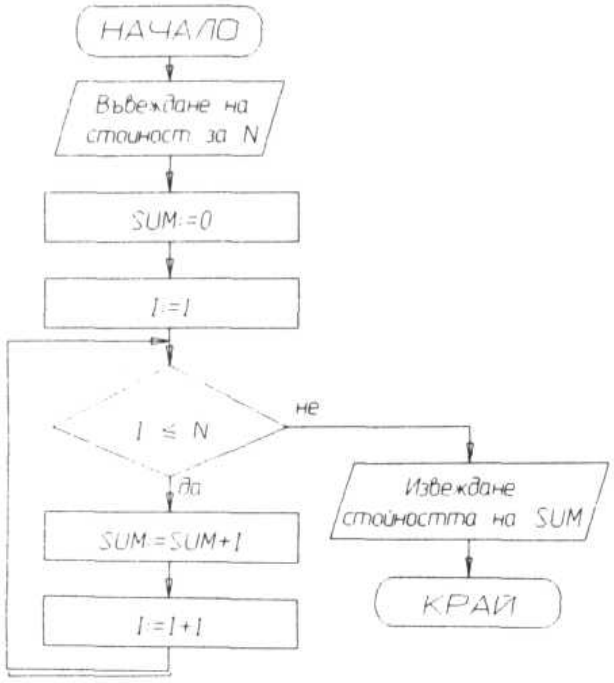
SUM - клетка от оперативната памет, в която ще се натрупва сумата на числата от 1 до N;

I - управляваща променлива на цикъла. Променливата I последователно приема всички целочислени стойности от 1 до N, с което "управлява" повторението на тялото на цикъла.

Реализация на алгоритъма на задачата като цикъл с постусловие е следната:



Същият пример, реализиран с цикъл с предусловие:



Видове цикли според броя на изпълненията:

Цикли с независим брой изпълнения

При този вид цикли броят на изпълненията на тялото на цикъла е известен предварително (преди започване на първото изпълнение на цикъла) и не зависи от междинните резултати, получавани в хода на изпълнението на цикъла.

Пример за такъв вид цикъл е алгоритъмът на задачата за намиране на сумата на първите N естествени числа.

Цикли със зависим брой изпълнения

При този вид цикли броят на изпълненията на тялото не е известен предварително и зависи от резултатите, получавани при всяко изпълнение на тялото на цикъла.

Най-често срещан случай на цикли със зависим брой - изпълнения са т.нар. итерационни цикли, при които с увеличаване на броя на изпълненията на тялото на цикъла се получават все по-точни приближения на търсения окончателен резултат. За всяко ново изпълнение на тялото се използват резултатите, получени от предишното изпълнение. Критерий за прекратяване на изчисленията е постигането на предварително зададена точност на резултата

6. Описание на алгоритъм чрез език за програмиране

Избор на език за програмиране

За написването на програма най-напред е необходимо да се избере езикът за програмиране, който ще се използва. Често този избор е направен още на етапа на съставяне на алгоритъма, тъй като степента на детайлизация на алгоритъма зависи и от възможностите на езика за програмиране, който се предполага да бъде използван.

От друга страна, съществуващите езици за програмиране предлагат определени специфични възможности, които ги правят подходящи за определен клас от задачи.

Съществуват разработени множество програмни езици.

Най-общата класификация на програмните езици е следната:

Машинно зависими: машинни; асемблерни.

Машинно независими: процедурно ориентирани; проблемно ориентирани.

Машинно-зависимите езици се разработват за всеки модел компютър (по-точно за всеки процесор) и представляват наборът от команди за този процесор.

**Машинни езици* - при тях командите на програмата (операциите и операндите към тях (или адресите на операндите в оперативната памет)) се записват направо в машинен код, което е изключително трудоемка работа, води неминуемо до грешки и прави тези езици практически неизползваеми;

** Асемблерни езици* - тези езици използват мнемоничен код - специални символи за обозначаване на командите. Мнемоничният код на асемблерните езици е значително по-удобен за използване от машинния код.

Машинно-независимите езици се разработват независимо от машинния език на компютъра, на който ще се използват и в този смисъл те стават универсални по отношение на начина си на използване. При тях програмата се описва на език, близък до говоримия човешки език (английски) и възприетата в математиката символика, а специална програма, наречена **транслатор** превежда текста на програмата, написана на такъв език, в машинния код на съответния компютър.

Процедурно-ориентираните езици* са в по-малка или по-голяма степен универсални по приложение езици. Използват се за решаване на широк кръг задачи от различни предметни области, както и за описание и публикуване на различни алгоритми, поради което те често се наричат и **алгоритмични езици. Такива са широко известните езици PASCAL, FORTRAN, BASIC, C, ADA.

**Проблемно-ориентираните езици* са езици с по-тясно и по-специфично приложение от процедурно-ориентираните езици - обикновено за решаването на задачи от една по-тясна предметна област.

Счита се, че колкото един програмен език е по-отдалечен от машинния, толкова той е от по-високо ниво.

7. Въвеждане, настройване и изпълнение на програма на компютър

На език от "високо ниво", какъвто е PASCAL, FORTRAN, BASIC, C, програмата се записва въз основа на съставения алгоритъм като последователност от оператори, които по начина на изписване наподобяват естествения човешки език, а формулите в програмите - тяхното представяне в математиката.

Съставената програма се нарича "изходен текст" или сорс код (source code) се въвежда в оперативната памет на компютъра. Но езиците от високо ниво (включително и асемблерните езици) са неразбираеми за компютъра. За да бъде изпълнена една програма на компютър, най-напред тя трябва да бъде "преведена" на машинен език (т.е. да бъде представена в машинния код на съответния компютър). Процесът на превеждане на програмата от език от "високо ниво" в машинен код се нарича **транслация**, а служебната програма, която извършва този превод се нарича **транслатор**. За всеки език от високо ниво и за всеки вид компютър, с който ще се използва този език, има разработен специален транслатор.

На практика се използват два вида транслатори - **интерпретатори и компилатори**.

Интерпретаторът работи по следния начин: поредният оператор от програмата се прекодира в последователност от елементарни машинни команди, които веднага се изпълняват от процесора; след това същото се извършва със следващия оператор и т.н. Не се получава цялостна програма на машинен език.

Компилаторът работи по следния начин: последователно се прекодират в машинен код всички оператори на програмата, получава се цялостна програма на машинен език, която се запомня (в оперативната памет или върху диск (дискета)) и може след това многократно да бъде изпълнявана.

Първият работещ транслатор е бил създаден през 1957 година за езика FORTRAN.

Най-общо процесът на въвеждане и изпълнение на програма, написана на език от високо ниво, може да бъде представен чрез следната схема:

- Въвеждане на изходния текст на програмата в оперативната памет;
- Транслация (коригиране на грешки) и получаване на програма на машинен език;
- Изпълнение на програмата - въвеждане на входни данни;
- Получаване на резултати;
- Тестване на програмата с различни входни данни.

Въпроси

1. Какви са основните етапи при подготовката и решаването на задача с компютър?
2. Какво е алгоритъм?
3. Какви са свойствата на алгоритъма?
4. Как се описват алгоритмите?
5. Как се описват графично алгоритмите?
6. Какви са базовите алгоритмични структури?
7. Представете базова алгоритмична структура верига.
8. Представете базова алгоритмична структура разклонение.
9. Представете базова алгоритмична структура цикъл.
10. Какви са видовете цикли според структурата им?
11. Какви са видовете цикли според броя на изпълнението?

12. Каква е класификацията на езиците за програмиране?
13. Какви са видовете транслатори?
14. Какви са въвеждане и изпълнение на програма, написана на Pascal?

Използвана литература:

- Чобанова Д., Turbo Pascal за начинаещи (части 1 и 2), изд. къща „Карисиа”, 1994
- Иванова Л., Вазова В., Танева Б., Стоянов С., Информатика за 9 клас, първа част, Коала прес, 2007.