Автори:
© Ангел Илиев Ангелов, 2002
© доц. Димитър Добрев Добрев, 2002
© Тошко Борисов Хиков, 2002

Рецензенти:
доц. Магдалина Тодорова, Факултет по математика и информатика на СУ
Анета Здравкова, учител по информатика, Софийска математическа гимназия

Редактор Красимир Гетов

Коректор Цветанка Манчева

Графичен дизайн:
© Анна Вълковска

Художник на корицата:
© Любомир Панов

СИЕЛА – СОФТ ЕНД ПЪБЛИШИНГ
София, 2002

ISBN 954-649-499-2

АНГЕЛ АНГЕЛОВ ДИМИТЪР ДОБРЕВ ТОШКО ХИКОВ

ИНФОРМАТИКА

ЗАДЪЛЖИТЕЛНА ПОДГОТОВКА

9. клас

Учебникът е одобрен
със заповед - РД 09 632/14.08.2002 г.,
на министъра на образованието и науката.

Оценители на съдържанието:
Юри Ангелов Карагьозов, Елена Тончева Драганова,
Мария Михайлова Христова, Дочка Панайотова Димитрова и
Пенка Йорданова Маринова;

Оценители на графичния дизайн
и здравно хигиенните изисквания:
Божидар Константинов Икономов, Петър Кръстев Добрев,
Николай Николов Стоянов, Регина Ванчова Далкалъчева и
Христо Николов Жаблянов.

ciela

СЪДЪРЖАНИЕ

1. НАШИЯТ ИНФОРМАЦИОНЕН СВЯТ

1.1. ИНФОРМАТИКАТА КАТО НАУКА

Като човешка дейност информатиката е стара „колкото света“ или поне „колкото човека“. Като наука тя започва да се обособява от сродните ѝ науки в средата на отминалия век, в тясна връзка с появилите се по това време компютри. Най-млад е терминът „информатика“. Като научно понятие той се появи едва в началото на седемдесетте години, а като елемент от нашето ежедневие – десетина години по-късно. Мощен тласък за повишения интерес към информатиката даде масовото навлизане на компютрите в практиката и в бита на хората. Впрочем, в много други страни за същото понятие се използва терминът ***computer science*** (наука за компютрите).

Информатика е дума от нашето съвремие и сигурно всеки има вече някаква своя – навярно доста добра представа – за нейния смисъл. От гледна точка на техническите ѝ средства – компютрите – тя стои някъде между математиката и електрониката. От гледна точка на приложенията обаче, информатиката първоначално се появява някъде там – между математиката, естествените науки и икономиката. Много скоро тя прониква в сферата на проектантската дейност, във всевъзможните офисни дейности и съобщенията, в полиграфията и издателската дейност и къде ли още не. В броени години, буквално пред очите ни, изчезнаха професии като тези на машинописките, чертожничките и словослагателите. Новите, които се появиха вместо тях, все още нямат своите утвърдени имена. Информатиката днес е проникнала толкова широко в нашето ежедневие, че неспециалистът започва да губи точна представа за фундаменталната същност и за основните очертания на тази бурно развиваща се област.

ИНФОРМАЦИЯ

Информатика произлиза от думата ***информация*** – латинското ***informatio***, което дословно означава ***„нещо формирано вътре, някаква представа“*** – представа за нещо от околния свят, която би ни помогнала по-добре „да се ориентираме“ в него.

Благодарение на своите сетивни органи и на умствените си способности ние можем да отразим в себе си свойствата на обектите, явленията и процесите от света, в който живеем.

Така, когато наблюдаваме, опипваме, измерваме обектите около нас, успяваме да придобием представа (да формираме знания) за тяхната ***форма***, ***големина*** и ***размери***, за ***цвета*** и ***теглото*** им, за ***количеството*** и ***местоположението*** им и т. н.

Когато наблюдаваме и изучаваме различните явления, като например ***топенето***, ***падането***, ***счупването***, ***преместването***, ***горенето*** и т. н., ние получаваме информация – придобиваме знания за различните свойства на обектите, които участват в тези явления.

Още повече информация получаваме, когато наблюдаваме протичащите край нас процеси. Като например: ***смяната на сезоните, растежа на животните, кръговрата на водата, разпространението на болестите, професионалното ориентиране на средношколците, законодателната дейност на парламента, жизнения цикъл на пчелните семейства, оборота на паричните средства в страната, енергопотреблението през зимата, компютризацията на обучението, разпространението на чуждиците, модните тенденции в козметиката, глобализацията в икономиката и в културата*** и т.н. Наблюдавайки ги, ние успяваме чрез съответен анализ да формираме представи и знания за участващите в тях обекти и субекти и за техните поведенчески свойства, както и за факторите, които движат тези процеси, за закономерността на причинно-следствените връзки и взаимоотношенията, които се наблюдават при тези процеси, както и за тенденциите на тяхното по-нататъшно развитие.

Всичко това хората правят, за да могат да предсказват бъдещето и по този начин да имат критерий за избор, винаги, когато трябва да решат как да постъпят в една житейска ситуация, в която могат да попаднат.

Естествено светът, в който живеем, е прекалено огромен и разнообразен, а поради това и необозрим за отделния човек. Езикът и способността ни за езикова комуникация са фундаменталното средство и подходът, които ни позволяват да обменяме информация и така да надскочим собствените си ограничени възможности. Като получаваме наготово знания, ние значително по-бързо и в по-широки гра-

ници придобиваме умения за целесъобразни действия. Като получаваме сведения от най-различни източници, придобиваме по-ясна и по-пълна представа за конкретната ситуация, в която реално се намираме, или в която можем да се окажем.

Информация се наричат сведенията, представите, знанията за обектите, явленията и процесите от околния свят – реален или въображаем, които хората формират вътре в себе си, или пък получават наготово от другите, и които те използват, за да могат да вземат целесъобразни решения при взаимодействието си с този свят.

Информацията е нещо неразривно свързано с човека и с неговото поведение. Фактът, че в своите действия той се ръководи от натрупаната в себе си информация, ни дава основание да определим поведението му като **информационно**.

Информацията би била немислима, ако не притежавахме органи, с които да възприемаме, запомняме и анализираме всичко от света около нас, както и да взаимодействаме с него. Такива възможности в една или друга степен притежават и много от животните. Интересно, а навярно и не случайно е, че с подобни органи и способности ние започнахме все по-често да надаряваме много от създаваните от нас технически творения. И наистина – доскоро такива бяха само компютрите. Днес подобни възможности притежават всевъзможни уреди и технически системи: като започнем от играчките и от битовата електроника, мобифоните и фотоапаратите, минем през кухненското обзавеждане и леките автомобили, през роботите, технологичните, транспортните и комуникационните системи, за да стигнем до самолетите и космическата техника.

ДАННИ

Видимо външно проявление на информационното поведение на хората е постоянното боравене с различни видове данни. Не бихме могли да си представим дори как би изглеждал нашият живот, ако не разполагахме с необходимите ни данни „за това и онова". Какво бихме правили, ако не разполагахме с вестници, радио, телевизия, със

справочници и речници, с указатели и каталози, с информационни бюлетини, с ръководства и нормативи и т. н.

И всичко това най-често е безплатно и общодостъпно за всеки. Но не винаги – има данни, които са изключително скъпи, като например патентите. Други – като секретните – са достъпни само за строго ограничен кръг от хора. Трети пък – така наречените документи, като например конституцията и законите, договорите, удостоверенията и др. имат правна сила. Много от тях са общовалидни и общодостъпни, но има и документи, като например банкнотите, които са временна собственост и дават права само на приносителя им. Има и такива, които са строго лични.

Така по закон всеки от нас е „изключителен собственик" на уникални данни.

Да погледнем за момент например личната карта на една ваша съученичка. Върху лицевата страна, както за всеки български гражданин, са вписани следните основни нейни ***лични данни***:

№ 161529649	РАДЕВА *RADEVA*	РАЛИЦА *RALITSA*	ДИМОВА *DIMOVA*
Пол Ж / F			ЕГН 8602043651
Гражданство	БЪЛГАРИЯ / BGR		
Дата на раждане	04.02.1986		
Валидност	27.02. 2012		

Фиг. 1

В тях е отразена важна лична и административна информация за притежателката на тази карта. За смисъла на повечето от елементите на тези данни ние, благодарение на житейския си опит, се досещаме и сами. За други можем да го узнаем и от стандартните надписи върху самата карта. За някои, както например за последните четири цифри на ЕГН, не знаем нищо.

Данните ***са смислово обособени порции информация, представени в някаква регламентирана форма, най-често писмена.***

Всеки такъв регламент уточнява състава, структурата и начина на **кодиране** (представяне, записване, произнасяне) на съответния

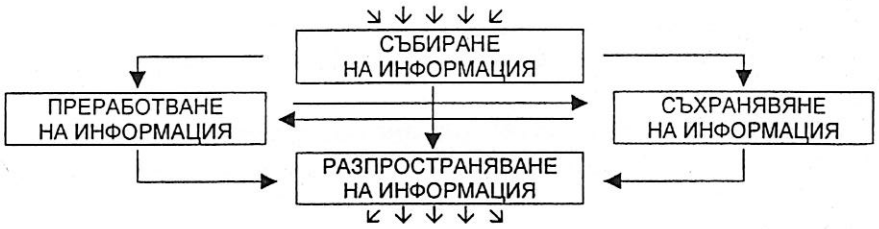
вид информация – това гарантира, че при нужда информацията ще може отново да бъде извлечена от тези данни.

Всички езици използват някакви специфични регламентирани форми за представянето на информацията като данни. Най-често използвани са писмените им форми. Важно изискване е те да бъдат удобни за различните видове дейности, които ще се налага да бъдат осъществявани с тези данни.

Поради езиковата им природа данните често се представят под формата на обикновен, последователен текст. Наличието на важни за практиката отношения между отделните компоненти на данните може да обуславя и друга структура на представянето им. Често явление в практиката са номерираните списъци, таблиците, изнесените под линия забележки, обособяването на общо съдържание, на терминологични индекси и показалци, на различни каталози и картотеки. Във всички случаи обаче, само онези, които познават кодировката, състава и структурата на наличните данни, биха могли да възстановят и оползотворят представената чрез тях информация.

ИНФОРМАЦИОННИ ДЕЙНОСТИ И ПРОЦЕСИ

Основни информационни дейности, присъщи на всички системи с информационно поведение, каквито са хората, някои технически системи, дори и животните, са **събирането**, **съхраняването**, **преработването** и **разпространяването** на информация. Следната схема показва информационните потоци, с които тези дейности са свързани функционално помежду си.



Фиг. 2

В своята взаимна връзка и в различни свои конкретни съчетания и последователности, информационните дейности образуват различни **информационни процеси**.

Така например обучението на едно дете е процес, в рамките на който то трупа знания от много и различни източници, като ги преосмисля и запомня, за да ги използва при собственото си поведение, както и да ги споделя със своите връстници, а и с всички, с които общува.

По подобен начин един компютър получава необходимата му управляваща информация (програма) и съответните данни за решаваната конкретна задача, за да получи чрез съответен анализ и синтез очакваното от потребителя решение.

В този процес обаче, като източник на данните и като потребител на резултатите участва и човекът. Т. е., човекът и компютърът се намират в своеобразна информационна симбиоза. Изходните данни, подадени от човека, се оказват входни данни за компютъра, и обратно – излъчените компютърни резултати са входни данни за човека.

Най-интересното обаче е, че същите те – човекът и компютърът, навярно в сътрудничество с други хора и компютри, с които се намират в подобни информационни взаимоотношения, биха могли да изпълняват заедно ролята например на една служба „Читателски заявки". Служба, която по същество е само елемент от един още по-сложен информационен процес, повтарящ отново общата схема на информационните дейности от фиг. 2, и реализиращ дейност, която бихме могли да наречем „Библиотечно обслужване".

КАК СЕ РАЖДА ИНФОРМАЦИЯТА

Най-прости случаи на „раждане" на информация са:

– **наблюдението,** съчетано с **регистриране** под формата на данни;

– **броенето**, съчетано с **регистриране** под формата на числа;

– **измерването**, съчетано с броене и **регистриране** под формата на числа.

По подобен начин се получава така наречената **първична информация**. Постигаме това чрез сетивата си, но често използваме и съответни технически приспособления и измервателни уреди като

термометри, линии, видеокамери.

Някои уреди са способни да правят същото напълно автоматично, без нашето участие. Например цифровите часовници и термометри, електромерите, цифровите волтметри, цифровите апарати за кръвно налягане и др.

Регистрирана по правилата на някакъв регламент, първичната информация се превръща в **първични данни**.

Първични данни с информация за настъпили събития са езикови съобщения от вида: *„Роди се дете." „Построиха мост." „Отсякоха дърво." „Пристигна колет." „Утвърдиха закон." „Беше измерена температура 17 °C." „Състоя се среща." „Дадоха положителен отговор на повдигнатия въпрос." „Определиха новия носител на Нобеловата награда в областта на физиката."* и др.

По-сложно, но от фундаментална важност, е получаването на нова информация чрез преработването на друга, вече съществуваща информация. Новополучената се нарича **вторична информация.** Получава се в резултат на анализ и синтез. Примери за това са: *закръгляването на число, сумирането на две числа, изчисляването на средна стойност, изчисляването на процент, редактирането на един текст, преводът на текст, изготвянето на конспект, изготвянето на списък на подалите молби за кандидатстване в средно специално училище, подреждането на списъка на кандидатите по бала, определянето на списъка на приетите, премахването от пълния списъка на онези, които не са приети.*

Чрез наблюдения и вторична преработка може да бъде изготвена **прогнозна информация** от вида: *„Утре ще вали, очаквани средни температури между 7 и 12 C°", „Вероятен победител е миналогодишният европейски шампион.", „Очаквани средни добиви за пшеницата – около 360 килограма на декар".*

Не се „ражда" нова информация, когато правим препис или копиране на съществуващ текст или документ. Нова информация би се съдържала единствено в съобщения от вида: *„Беше направено ново копие на документа", „Копието на документа бе направено на 25 март 2002 г.", „Копие от документа получи и ищецът", „Копие от документа бе връчено на ответника срещу подпис."* И всичко това, разбира се, ако някой е констатирал съответния факт и го запомнил, записал

Възможна е и „смърт" на информация – неволна или преднамерена, както ако едно писмо е изгоряло или пък ако е било изгорено, или ако изтрием дискета с някакви данни. Защита срещу подобна „смърт" можем да постигнем чрез своевременното копиране на документа или дискетата, или пък чрез „заключване" на дискетата срещу запис. Загубването на единствен екземпляр от някакви данни е фактически „смърт" за тях, а повторното им намиране – истинско „възкресение".

Освен чрез първична регистрация и чрез преработка, информация може да бъде получена наготово. **Източник на информация** в този смисъл може да бъде човек или техническа система, а формата – данни, предадени като **съобщение.** Съобщението може да бъде предадено гласово, но твърде често е записано върху някакъв траен материал – **носител на информация** (хартия, дискета и др.).

В този смисъл справочниците и библиотеките също са източници на информация.

ВИДОВЕ ИНФОРМАЦИЯ

В зависимост от своята смислова същност и от своето функционално предназначение, информацията може да бъде **числова** (**количествена**), **идентифицираща** (**именуваща**), **календарна**, **адресна**, **описателна**, **експлоатационна**, **правна** и др. В говоримия език твърде често подобни същностни и функционални особености на различните видове информация се отнасят и към самите данни, които представят тази информация.

В зависимост от своя вид и структура, данните представящи една информация или отделни нейни компоненти, могат да бъдат **цифрови**, **буквени**, **пиктограмни**, **нотни**, **щрихови**, **графични**, **геометрични**, **растрови**, **текстови**, **списъчни**, **таблични**, **картотечни** и др. В говоримия език твърде често подобни външни особености на формата и структурата на данните се отнасят и към самата информация, която тези данни представят.

В личната карта на фиг. 1 например са включени:

– един номер;

– четири имена (три лични и едно на държавата ни) – всяко кодирано по два различни начина;

дирано по два различни начина;

– един определител за пол – пак в две кодировки;

– един десетцифров код (неправилно наричан номер);

– шест неизменяеми коментарни пояснения, от които № е символ, произлязал от френското *„numéro"*, а ЕГН е абревиатура (съкращение) на *„Единен граждански номер"*.

ИНТЕРПРЕТАЦИЯ НА ДАННИТЕ

Едни и същи данни могат да бъдат използвани по различни поводи. Така в конкретен списък за лица с техните лични данни, някой може да търси свой съименик, друг – свой връстник, трети – съгражданин. Може да ни интересува общият брой на лицата или броят на хората в пенсионна възраст, или пък възрастовият диапазон на мъжете, включени в списъка, и т. н.

Съществува обаче и едно друго различие. То не е присъщо на данните сами по себе си, а е породено от нашето моментно отношение към тях. Водоразделът се определя от това дали данните се формират в момента, или се използват данни, формирани по-рано. Във втория случай е интересно и за какво точно се използват – като информационна суровина, като критерий за избирателно действие или като указания, които пораждат и управляват някакви други, още по-сложни процеси. Тълкуваме данните **обектно**, когато ги създаваме, или когато ги ползваме като информационна суровина – т. е. **фактологично** – за анализ и синтез. Тълкуваме данните **субектно**, когато ги използваме като критерий за избор, или пък – **процедурно** – като указания и предписания за действия. Възможно е обаче едни и същи данни, по различни поводи, да бъдат **интерпретирани** (тълкувани) по няколко от възможните начини. Например номерацията в един списък от хора може да бъде използвана, за да узнаем колко ***фактически*** е техният брой, или пък, за да се определи ***процедурата*** (последователността), по която хората ще бъдат обслужвани. Една и съща лекарска рецепта например се тълкува ***обектно*** – в процеса на нейното съставяне, ***фактологично*** – когато се преписва или изучава, и ***процедурно*** – когато се изпълнява в аптеката. Пак ***процедурно***, но по друг начин, тя се тълкува от болния, когато той следва отразените в нея лекарски предписания.

ЕЗИЦИ

За да можем да получаваме и да предаваме информация, да я съхраняваме, и да я използваме отново при нужда, да я разчленяваме или обединяваме, както и да я преподреждаме, е необходимо да разполагаме с начин за нейното материално, трайно (овеществено чрез образи и знакове) представяне.

Както вече знаем, това е възможно благодарение на факта, че информацията може да се представи под формата на данни. Тази форма има езикова същност и природа. Всеки **език** представлява система от две основни компоненти: азбука и граматика, която от своя страна включва синтаксис и семантика. **Азбуката** определя допустимите в езика **знакове** – някакво крайно множество от различими елементи (звукови, графични, електромагнитни, дупкови, отражателни и др.). Правописните правила – **синтаксисът** на езика, определят допустимите начини за съчленяване на знаковете в по-сложни конструкции, като например думите, изреченията и текстовете или формулите, таблиците. Смисловите правила – **семантиката**, определят съответствието между езиковите конструкции и информацията, която те представят. Именно чрез установяването на съответствие между различните езикови конструкции и формираните от нас представи, езикът се превръща в изразно средство за представяне на информация.

Съответствието между допустимите в езика знакове и конструкции и смисъла, вложен в тях, е въпрос на чисто договаряне и на приемственост между хората, които използват този език. Общоизвестно е например, че смисълът, който ние влагаме в кимането с глава, използвано като знак за потвърждение или отрицание, е обратен на този, който практикуват болшинството други народи. Друг подобен пример е знакът „–“, който в езика на математиката означава операция за намиране на противоположната стойност или пък операция за изваждане, в българския език (и не само в него) служи за разделителното тире, докато в телеграфната азбука на Морз е просто единият от двата използвани в нея знакове (. и –) и няма самостоятелна употреба и смисъл.

Освен естествените езици в практиката се използват и много специализирани езици. Различията при тях твърде често са по-голе-

ми и са обусловени от спецификата на съответните области на приложението им. Примери за това са езикът на математическите формули, езикът на пътните знакове, езикът за записване на шахматни партии, нотните езици. Познаваме езици, използващи пиктограми – схематични рисунки, поясняващи например начина на пране и гладене на дрехи, или начина на товарене, подреждане и съхраняване на кашоните, върху които тези знакове са нанесени и т. н.

Езиците като явление са възникнали заедно с хората, като са съпровождали тяхното развитие още от най-ранните стъпки на тяхното развитие.

Първоначално, както и при животните, езиците служели за любовни послания и за предупредителни съобщения при опасност, както и за съгласуване на най-простите действия при групово придвижване и ловуване. По-късно, и то само при някои животински видове, започнали да служат за предаване и на по-сложни съобщения. Само при човека те поели и изключително важната роля да служат и като „овеществена“ или по-точно „ословесена“ форма за мисловна дейност. Всъщност ние мислим чрез термините на родния си език. Неопровержим пример за това е смятането. Можем веднага да се убедим в това, ако опитаме да умножим две числа, но „на глас“ и то на някой чужд език.

Със своите писменни форми езикът поема и още една изключително важна функция – тази на допълнителна, дългосрочна памет, надхвърляща рамките на един човешки живот. Защото веднаж записана, една информация винаги може да бъде прочетена отново.

СЪЩНОСТ НА ИНФОРМАТИКАТА

Информатиката е млада наука. Нейното истинско и богато съвременно съдържание – и като теория, и като практика – е свързано с появата, развитието и приложението на съвременните компютри. Това е причината, поради която в днешно време терминът информатика се отъждествява, и то с основание, с понятието **компютърна информатика.**

Науките се раждат, развиват и усъвършенстват заради своите приложения. И математиката, а особено информатиката (като по-млада) включват в кръга на предметната си област и проблемите на кон-

кретните си приложения в различните области на практиката.

Предмет на **компютърната информатика** са:

– средствата и методите за представяне на информацията като данни;

– структурните форми за представянето на информация с различно функционално предназначение, както и методите за техния анализ и синтез;

– техническите системи (компютрите) и тяхното програмно (софтуерно) осигуряване, позволяващо автоматична работа с данни;

– методите и правилата (алгоритмите) за автоматизирано осъществяване на дейностите по събиране, съхраняване, преработване, разпространяване и пренасяне на данни;

– методите, средствата и технологиите за разработване, внедряване и поддържане на общи и специални програмни системи, ориентирани към всевъзможните приложения на компютрите;

– усъвършенстване и по-нататъшно развитие на собствения теоретичен фундамент.

ВЪПРОСИ И ЗАДАЧИ

1. Символ на какво е знакът → и какъв е неговият произход?
2. Абревиатура на какво са BG, BGR, БНР, БНТ, БНБ?
3. Посочете примери за засекретени данни?
4. Какви графични означения познавате за числата от 1 до 6?
5. Определете смисъла на следните графичните символи:

6. Информация за какво се съдържа в числовата данна 3,14?
7. Каква информация се съдържа в изречението:

"Как е леко и бързо изчислено пи всеки знае щом желае"?

8. Документ ли е автобусният билет и каква е неговата правна сила?
9. Документ ли е тото-фишът и каква е неговата правна сила?

Каква първична информация може да се извлече от него?

Каква вторична информация може да се извлече от него и какви допълнителни данни са необходими за това?

Да се заслушаме в следния разговор между майка и дъщеря:

– Колко съученици ти дойдоха на гости?

– Не забелязах, но веднага ще ги преброя: един, двама, трима, четирима – всичко пет.

– Добре. Вземи, моля те, пет ябълки; сложи и още по една – за нас двете.

– Пет плюс две? Ами значи – общо седем.

Ей сега, ето ги: една, две, три, четири, пет, шест, седем.

Зад този делничен и малко наивен розговор е скрита едва ли не цялата същност на математиката.

Броенето, числото, записването и боравенето с числа, са едно от великите – фундаменталните „изобретения" на човечеството, така фундаментално, както речта, словото, писмеността – всичко онова, което наричаме език. Впрочем, тези две „изобретения", възникнали много, много отдавна, са навлезли така дълбоко в битието на човека, че са станали наша втора природа – толкова естествена, че като че ли не се нуждае от никакви обяснения. Науката винаги има обаче какво да ни поднесе, и като правило – все в наша полза. Научно обяснение за същността на числата и техните представяния можете да намерите в края на урока.

МАЛКО ИСТОРИЯ

Една от най-старите археологически находки, която безспорно свидетелства за първите количествени представи на хората, е открита в Моравия и е на възраст около 30 000 години. Представлява кост от животно с отбелязани върху нея 55 черти, подредени в два реда и групирани по пет. В тази малка подробност няма нищо изненадващо, защото пръстите на ръцете са били първото и естествено човешко „сметало". В този смисъл закономерно е, че още от самото

начало като означения за числа са били използвани пиктограмите (рисунковите знакове) за пръст, ръка и две ръце, превърнали се в йерогилифи, а още по-късно и в знаковете **I, V** и **X**, които ползваме и до днес. Не е случайно също и езиковото родство между думата ***пет*** и думите ***педя, петура, пестник.***

Първите пиктограми за числа са се появили в Египет преди повече от 5 000 години. Различни древни народи като шумерите, маите, китайците и др. също изобретили по различно време подобни писменни знакове. Древните гърци използвали за тази цел чертата и първите букви от думите за пет, десет, сто, хиляда, а по-късно – буквите от собствената си азбука, всяка с допълнителен щрих, подсказващ особения ѝ смисъл като число. По подобен начин са били използвани и буквите на нашата кирилица.

Всички тези форми били удобни най-вече за съхраняването в писмен вид на количествени данни, както и за записването на календарни дати. Например годината 2001-ва, записана с някои древни бройни системи, изглежда така:

Стремежът към рационализиране на дейностите с числа довежда до идеята за тяхното позиционно представяне – същото, което ползваме и до днес. Възникването му се отнася към IV в. преди новата ера, когато в Индия започнали да си служат с десетични числа. Век по-късно в Месопотамия се появява и нулата, с което позиционната идея добила по-завършен вид. Така възникнали формите за представяне, а заедно с тях и методите за пресмятане с числа, които използваме и до днес. Основно предимство на позиционните системи е простотата на аритметичните операции с числа, представени по този начин. Всяко съвременно дете изучава правилата за операции с числа още в първите години на своето обучение. Широкото разпространение на тези методи дължим на трудовете на арабския математик и астроном Мохамед Ибн Муса ал-Хорезми, написани през първата половина на IX в. Тези методи реално стават достояние на европейците едва през XIII в. Истинско, широко разпространение те получават обаче едва на границата между XVI и XVII в. Невероятно е, че за това са били необходими повече от седем века, и че дълго време са срещали значителна съпротива.

ТЕХНИЧЕСКИ СРЕДСТВА ЗА ДЕЙСТВИЯ С ЧИСЛА

Числата не съществуват реално в природата. Те са хитроумна, но нематериална (**абстрактна**) човешка измислица. За да може да ги онагледява (материализира) – и при броене и при смятане – човекът се нуждаел от някакви помощни средства. Първоначално приспособил неща, които намерил наготово в природата. По-късно започнал и сам да изобретява такива. Така, на различни стадии от своето развитие, той използвал или създал за същата цел следните изчислителни средства:

– **пръстите на ръцете** – естествено и винаги подръчно средство, облекчаващо броенето, събирането и изваждането;

– **ра́бошът** – дървена пръчка, върху която количествата се отбелязват с резки; разцепена на две, тя лесно се превръща в двойка идентични „документи" – по един за кредитора и за длъжника;

– **броеницата** – облекчава броенето, събирането и изваждането;

– **аба́кът** – познатото на всички сметало, основано на десетичното представяне на числата – облекчава четирите аритметични операции;

– **линията** – достатъчно дълга рейка, предварително разграфена и надписана – облекчава измерването на дължини, като го свежда до проста, тритактова процедура, включваща налагане и отчитане;

– **справочната таблица** – съдържа готови, предварително изчислени резултати,

– **логаритмичната (сметачна) линийка** – изобретена от Джон Непер, основана на операциите с логаритмични стойности – облекчава умножението, коренуването и някои други изчисления;

– **сметачните машини** – механични изчислителни устройства с ръчно управление:

- първи проекти – Леонардо да Винчи (втората половина на XVI в.) и Вилхелм Шикард (1623 г.);
- първа действаща машина – Блез Паскал (1641 г.);
- първа реализация и на четирите операции – Готфрид Лайбниц (1673 г.);
- първо серийно производство на сметачни машини – Карл Томас (1821 г.);

– **електронните калкулатори** – електронни изчислителни устройства с ръчно управление, появяват се масово в края на 60-те години на XX век;

– **компютрите** – автоматични изчислителни системи с програмно управление:

- първи механичен проект – Чарлз Бабидж (1834 г.),
- първи електронни двоични схеми – Джон Атанасов (1936 – 1941 г.);
- първи действащ компютър – Джон Екърт и Джон Моучли (1946 г.);
- серийно производство на компютри (от 50-те години на XX в.);
- производство на персонални компютри (от 80-те години на XX в.).

СЪЩНОСТ НА ПОЗИЦИОННИТЕ СИСТЕМИ

Добре познатата ни **десетична позиционна система** е всъщност само един от възможните начини за **кодиране** на числата. Нарича се **десетична**, защото използва само десет различни писмени знака (цифри), а е **позиционна**, защото всяка използвана цифра „тежи" различно, според мястото (позицията), в която е употребена при представянето на едно число.

Всъщност една цифрова последователност, като например **28104**, е само съкратеният запис на петчленната сума

$$\begin{array}{rr} & 20000 \\ & 8000 \\ + & 100 \\ & 00 \\ & 4 \\ \hline & 28104 \end{array}$$

или на израза **20000 + 8000 + 100 + 0 + 4**. Това е начинът, по който лесно можем да възстановим стойността на числото. Впрочем, това е и начинът, по който произнасяме същото това число.

Друга, по-математизирана форма на този израз, е степенният многочлен: $2.10^4 + 8.10^3 + 1.10^2 + 0.10^1 + 4.10^0$. В него още по-ясно личи особената роля на числото **10**, послужило като **основа**, а от там и като наименование на десетичната позиционна система.

Обобщена за десетично число с **k+1** цифри, тази формула придобива вида:

$$N = ak.10^k + a_{k-1}.10^{k-1} + \ldots + a_1.10^1 + a_0.10^0.$$

Нека изрично напомним, че навсякъде тук коефициентите a_0, a_1, ... a_k са цели числа от интервала **[0, 9]** – същите, чиито цифри използваме в цифровата последователност **ak ak-1 ak-2 ... a1 a0**, представяща съкратения запис на числото **N .**

Най-голямата ценност на десетичната позиционна система, а и на позиционните системи въобще, се крие в простотата на правилата за осъществяване на аритметичните операции с числа.

Когато по-късно в математиката се появили и десетичните дроби, обобщение претърпяла и десетичната позиционна система. С въ-

веждането на десетичната запетая станали възможни съкратени записи от вида **127,537**, обединяващи в един запис числото с неговата цяла и дробна част. Стойността на числото в такъв случай се изчислява по формула, която използва и отрицателни степени на основата **10**.

$$127{,}537 = 1.10^2 + 2.10^1 + 7.10^0 + 5.10^{-1} + 3.10^{-2} + 7.10^{-3} =$$

$$= 100 + 20 + 7 + \frac{5}{10} + \frac{3}{100} + \frac{7}{1000}$$

ДРУГИ ПОЗИЦИОННИ СИСТЕМИ

Освен **десетичната**, историята помни и други бройни системи за представяне на числата, като например **дванадесетичната** (жива следа от която е броенето с дузини) и **двайсетичната** – каквато практикували маите. Следи от двадесетична система откриваме например и във френското 99, което се пише и произнася като quatre-vingt-dix-neuf, т. е. 4 x 20+10+9. **Шейсетична** по същество е системата, която практикуваме всички, когато мерим ъгли или времена в минути и секунди (съответно ъглови или за време). И макар че не разполагаме със самостоятелни писмени знакове за всички тези позиционни системи, математическите им принципи неизменно остават едни и същи. Единствената разлика е в числото, което сме използвали като основа на съответната позиционна система.

Всъщност всяко цяло число **b**, по-голямо от **1**, би могло да изпълнява ролята на такава основа. Естествено показаната по-горе математическа формула, също ще претърпи обобщение и ще придобие вида:

$$N = a_k.b^k + a_{k-1}.b^{k-1} + \ldots + a_1.b^1 + a_0.b^0$$

където коефициенти a_0, a_1, ... a_k са числа от интервала **[0, b-1]**. Именно за тях ще ни потрябва подходящ набор от цифри – толкова на брой, колкото е избраната основа.

ДВОИЧНА ПОЗИЦИОННА СИСТЕМА

Основа на двоичната позиционна система е числото 2, а като цифри се използват само знаковете **0** и **1**.

В съвременните компютри всички данни се записват и съхраняват под формата на числови кодове, представени в двоична позиционна система. Причина за това е простотата и надеждността на техническите средства, с които вътре в компютърната памет се имитират двоичните цифри **0** и **1**, както и простотата на електронните схеми, реализиращи операциите с тях.

Изчисляването на стойността на едно число, записано в двоична позиционна система, се осъществява по формулата:

$$N = a_k.2^k + a_{k-1}.2^{k-1} + \ldots + a_1.2^1 + a_0.2^0$$

Например: $01101_{(2)} = 0.2^4 + 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 =$

$= 0.16 + 1.8 + 1.4 + 0.2 + 1.1 = 13_{(10)}$.

Тук ясно личат „теглата" на последователните позиции в двоичното представяне на числата. Естествено това са последователните степени на основата **2**. Ясно е, че и тук, както в десетичната система, нулите отляво са незначещи. Ясно е също, че например **1101** не е числото *„хиляда сто и едно"*. За да се предпазим от подобни заблуди, в скоби е указана използваната позиционна система.

Таблица 1 показва всички числа, които могат да се запишат с четири двоични цифри.

$0_{(10)} = 0000_{(2)}$	$4_{(10)} = 0100_{(2)}$	$8_{(10)} = 1000_{(2)}$	$12_{(10)} = 1100_{(2)}$
$1_{(10)} = 0001_{(2)}$	$5_{(10)} = 0101_{(2)}$	$9_{(10)} = 1001_{(2)}$	$13_{(10)} = 1101_{(2)}$
$2_{(10)} = 0010_{(2)}$	$6_{(10)} = 0110_{(2)}$	$10_{(10)} = 1010_{(2)}$	$14_{(10)} = 1110_{(2)}$
$3_{(10)} = 0011_{(2)}$	$7_{(10)} = 0111_{(2)}$	$11_{(10)} = 1011_{(2)}$	$15_{(10)} = 1111_{(2)}$

Таблица 1

За представянето на числа, по-големи от $15_{(10)}$, ще бъдат необходими по-дълги двоични кодове.

Най-ранни следи от идеята за двоичната система се откриват в непубликуваните трудове на Томас Хериот от 1605 г. Век по-късно – в 1703 г., немският математик Лайбниц пръв публикува принципите на двоичното смятане. Първото електронно-лампово изчислително устройство, основаващо се на двоичната система, е проектирано през

1939 г. от американския физик от български произход Джон Атанасов, в сътрудничество с Клифърд Бери. Оттогава всички компютри работят на този принцип.

ПРЕВОД НА ЧИСЛА ОТ ДВОИЧНА В ДЕСЕТИЧНА СИСТЕМА

Както стана ясно по-горе, един съкратен двоичен запис (код) може да бъде осмислен като число чрез формулата:

$$N = a_k . 2^k + a_{k-1} . 2^{k-1} + \ldots + a_1 . 2^1 + a_0 . 2^0$$

Например: $1101_{(2)} = 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 = 1.8 + 1.4 + 0.2 + 1.1 = 13_{(10)}$.

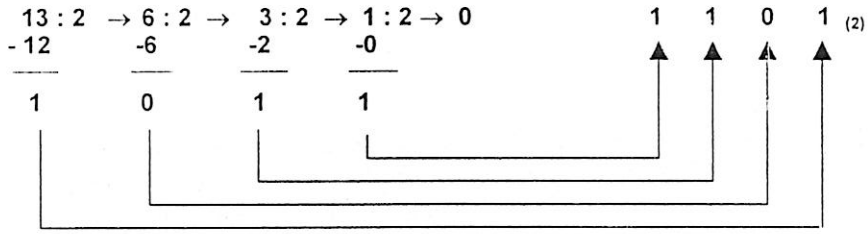
Тук ясно личат „теглата" на последователните позиции в двоичното представяне на числата. Естествено това са последователните степени на основата **2**. В таблица 2 са дадени в готов вид стойностите на някои от тях:

$2^0 = 1$	$2^4 = 16$	$2^8 = 256$	$2^{12} = 4096$	$2^{16} = 65536$	$2^{20} = 1048576$
$2^1 = 2$	$2^5 = 32$	$2^9 = 512$	$2^{13} = 8192$	$2^{17} = 131072$	$2^{21} = 2097152$
$2^2 = 4$	$2^6 = 64$	$2^{10} = 1024$	$2^{14} = 16384$	$2^{18} = 262144$	$2^{22} = 4194304$
$2^3 = 8$	$2^7 = 128$	$2^{11} = 2048$	$2^{15} = 32768$	$2^{19} = 524288$	$2^{23} = 8388608$

Таблица 2

ПРЕВОД НА ЧИСЛА ОТ ДЕСЕТИЧНА В ДВОИЧНА СИСТЕМА

Съществува правило, което би ни позволило да преведем всяко желано цяло число от десетична система в двоична. Ще поясним неговото действие чрез конкретен пример. Следващата схема показва превода на числото $13_{(10)}$.



Преводът на числото се осъществява чрез многократно деление с основата **2**. Като начално делимо използваме самото превеждано число. Получаваните на всяка стъпка остатъци формират двоичното представяне на числото, но от дясно на ляво. Целите резултати от делението пък продължават да участват като делими в следващите стъпки. Процесът се прекратява, когато резултатът от делението стане **0**.

ЕДНА ДРУГА УДОБНА ФОРМУЛА ЗА ПРЕВОД НА ЧИСЛА

Най-удобна като изчислително правило за превод на числа от една позиционна система в друга е **формулата на Хорнер**

$$N = ((\ldots (a_k . b + a_{k-1}) . b + \ldots) . b + a_1) . b + a_0$$

Получена е от познатия ни вече степенен многочлен чрез подходящи еквивалентни преобразувания, като многократно сме изваждали общия множител b след скоби:

$$\begin{aligned} N &= ak.bk + ak\text{-}_1.bk^{-1} + \ldots + a_1.b^1 + a_0.b^0 = \\ &= ak.bk + ak\text{-}_1.bk^{-1} + \ldots + a_1.b + a_0 = \\ &\quad (ak.bk^{-1} + ak\text{-}_1.bk^{-2} + \ldots + a_{1)}.b + a_0.b^0 = \\ &\quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &= (\ldots (ak.b + ak\text{-}_1).b + \ldots + a_1)b + a_0.b^0 \end{aligned}$$

Ще използваме тази формула във вариантаи за $b = 2$. Така преводът например на числото $1101_{(2)}$ придобива вида:

$$1101_{(2)} = ((1.2 + 1).2 + 0).2 + 1 = 13_{(10)}$$

Формулата е удобна, защото позволява да получим числената стойност на едно число чрез еднократен „прочит" на цифрите му от ляво на дясно, без да е необходимо да изчисляваме или помним последователните степени на основата и да внимаваме кога и коя от тях да използваме.

По отношение на обратния превод формулата не дава нищо ново, но именно тя обяснява действието на правилото, което показахме наготово по-горе. И наистина, щом като например $13_{(10)}$ = **((1. 2 + 1). 2 + 0). 2 + 1**, то ако за момент означим израза **((1. 2 + 1). 2 + 0)** с **p**, ще получим $13_{(10)}$ = **p . 2 + 1**. Но от това веднага става ясно, че ако разделим превежданото число $13_{(10)}$ на основата **2**, ще получим остатъка **1** и цяла част **6**, която всъщност е равна на все още неразчлененатa стойност на израза **(1.2 + 1).2 + 0)**. Продължавайки многократно по същия начин, ще получим една по една всичките цифри на двоичното представяне на числото, но в техния обратен ред.

НЯКОИ СТРАННИ ПОЗИЦИОННИ СИСТЕМИ

Макар и рядко – за означаване на часове и месеци – все още се използват така наречените римски цифри **I**, **V** и **X**. Римска се нарича и съответната система за записване на числата. Следната таблица показва числовите съответствия на един малко по-широк набор от римски цифри.

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

Ако се вгледаме по-внимателно, бихме могли да открием и в римската система някакъв позиционен ефект, защото и тук значението на една цифра зависи от позицията, в която тя се намира. Така например **IV** и **VI** съдържат едни и същи цифри, но употребено отляво, **I** намалява стойността на **V** с единица, докато в случая, когато е отдясно, я увеличава. Това правило е приложимо за **I** спрямо **V** и **X**, за **X** – спрямо **L** и **C**, а също и за **C** – спрямо **D** и **M**.

Така например числата от 39 до 50 могат да се запишат, както следва:

39	40	41	42	43	44	45	46	47	48	49	50
XXXIX	XL	XLI	XLII	XLIII	XLIV	XLV	XLVI	XLVII	XLVIII	XLIX	L

Нека обаче спрем дотук и без да си задаваме въпроса как се събират или умножават подобни числа, с облекчение и благодарност да си спомним за древните народи, които са ни завещали десетичната позиционна система.

Впрочем римската не е единствената странна позиционна система, която използваме. Нека се опитаме да запишем с числа часа и датата секунда преди настъпването на новото хилядолетие. Ще получим навярно нещо от вида: 23:59:59 31.12.2000. Но как само секунда по-късно този запис ще се превърне в 00:00:00 01.01.2001? Можем да отстраним някои от неговите странности, като го препишем във вида 2000.12.31 23:59:59 или даже като 20001231235959, с което ще осигурим възможност преносите да се изпълняват винаги наляво. Но кога настъпват тези преноси? Въпреки че сме премахнали всички разделители и използваме само десетични цифри, това не е десетична система. Причината са неравноделните интервали за секунди, минути, часове, денонощия, месеци и години, с които традиционно мерим времето. И наистина 60 минути образуват един час, 60 секунди образуват една минута, а 24 часа образуват едно денонощие. Но колко денонощия правят един месец? За да се справим, ще ни потрябват доста специални знания, включително и кокалчетата на двете ни ръце.

Все пак и това, което постигнахме, е нещо полезно, защото поне ще можем лесно да проверим дали например 20001231235959 е по-малко от 2 0010101000000. Достатъчно ще бъде да погледнем на тези кодове като на числа в десетична система. Облекчен вариант на тази идея, но за същата цел, е използван по отношение на първите 6 цифри на нашите единни грждански номера (ЕГН).

ЗА ПОЗНАТИТЕ НЕЩА – С ЕЗИКА НА НАУКАТА

Броенето е дейност – многоетапен процес, чрез който определяме количествената характеристика на едно множество, т. е. получаваме отговор на въпроса: *„Колко са елементите в това множество?“*.

Числото е мярка за количество – отговор на въпроса *„колко?“*, получен чрез броене. Числото е абстрактен образ на едно количество, защото от него (само по себе си) не личи какви са елементите с установения брой. За това при нужда се налага да правим допълнителни уточнения (пет пръста, пет ябълки, пет лева и т.н.). В този смисъл така наречените **естествени числа** (нула, едно, две, три, ...) са универсални мерки за количество.

Бройните системи служат за представяне на числата със знакове или с имена. Едно и също число може да бъде представено изобщо по много различни начини. (Например: ∴, IIIII, ИІ, ✋, V, 5, 🕔, пет, пять, πεντε, five, cinq, fünf ...).

Най-разпространени са бройните системи, използващи цифри.

Цифрите всъщност са само средство – графични знакове, за означаване на числа. Всяка цифра означава точно определено число. Цифрите имат и свои словесни аналози (думи, които означават същите числа). Поради това, че числата са безкрайно много, не е възможно за всяко число да има отделна цифра и дума.

Позиционните системи са особен вид бройни системи, конструирани така, че да облекчават операциите с числа. Най-широко разпространена е десетичната система. Тя е узаконена и в Международната система за мерните единици – **SI**.

Отброяването е дейност, обратна на броенето. Осъществява се пак като броене, но целта е друга – като знаем едно число, да отделим толкова елемента или да повторим някаква дейност толкова пъти, колкото е това число, т. е. на едно абстрактно число да съпоставим отново съответното количество реални неща (не непременно същите, каквито първоначално сме преброили).

Основни дейности с числа са **регистрирането** (броенето) и **интерпретирането** (отброяването), както и познатите ни **аритметични операции**: събиране, изваждане, умножение, деление, степенуване и др., които дават отново числови резултати.

Преводът на число е дейност, изразяваща се в замяната на едно представяне на числото с друго, без това да променя неговата **стойност** (представеното чрез числото количество).

Най-накрая нека да разсеем една заблуда, свързана с термините *двоично* или *десетично число,* или пък *римско* или *арабско число.* Истината е, че такива числа няма. Числата изразяват количества и поради това не могат да бъдат нито двоични, нито десетични, нито римски, нито арабски. Така че, ако използваме подобни термини, то е само като по-кратък и интуитивно ясен математически жаргон, имащ отношение не към числата, а към техните конкретни представяния.

ВЪПРОСИ И ЗАДАЧИ

1. Каква бройна система използваме при взаимодействието си с електронните калкулатори?
2. Какви бройни системи се използват по циферблатите на часовниците?
3. Какви бройни системи се използват за означаване на календарните дати?
4. Кое е най-голямото число, което се записва с три десетични цифри?
5. Кое е най-голямото число, което се записва с три двоични цифри?
6. Преведете в десетична система числото $101_{(2)}$.
7. Преведете в двоична система числото $101_{(10)}$.
8. Намерете и сравнете помежду им двоичните представяния на числата $30_{(10)}$, $31_{(10)}$, $32_{(10)}$ и $33_{(10)}$.
9. Изчислете следните изрази :
   a) $0110_{(2)} + 0010_{(2)}$
   b) $0101_{(2)} + 0100_{(2)}$
   c) $0100_{(2)} + 0011_{(2)}$
   d) $1101_{(2)} - 1011_{(2)}$

e) $1110_{(2)} - 1101_{(2)}$

f) $1111_{(2)} - 1011_{(2)}$

g) $0100_{(2)} \times 1100_{(2)}$

h) $0101_{(2)} \times 1010_{(2)}$

Упътване: Преведете числата в десетична система, изпълнете съответната операция, преведете резултата отново в двоична система.

10. Колко различни петцифрени комбинации от 0 и 1 са възможни?

11. Попълнете докрай следната таблица за многоцифрените представяния на числа в двоична позиционна система:

k *- брой на цифрите в двоичния код*	1	2	3	4	5
n *- брой различни комбинации от* ***k*** *двоични цифри*	2	4	8		
z *- най-малкото число, представимо с* ***k*** *двоични цифри*	0	0	0		
m *- най-голямото число, представимо с* ***k*** *двоични цифри*	1	3	7		

12. Опитайте да намерите формулите, по които могат да бъдат изчислени стойностите на n и m от горната таблица.

13. Колко различни числа могат да се запишат с осем двоични цифри и кое е най-голямото от тях?

14. Кое е най-голямото число, което може да бъде записано с десет двоични цифри?

15. Кое е най-голямото число, чието двоично представяне съдържа 2 нули и 2 единици?

16. Изчислете кой пореден ден от съответната календарна година е бил вашият рожден ден и представете съответното му число в двоична позиционна система.

17. Изберете едно конкретно естествено число m.

a) Намерете двоичното представяне на m.

b) Добавете в края му още една **0**.

c) Намерете числото n, което съответства на получения в b) двоичен код.

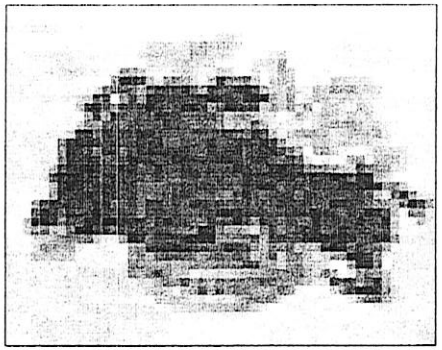
d) Вярно ли е, че $n = 2m$?

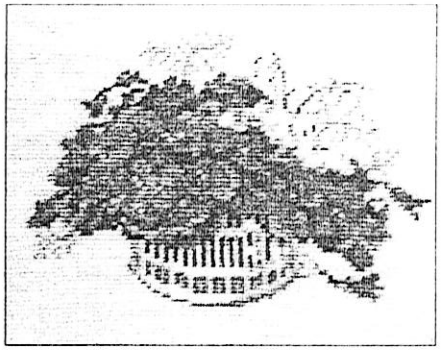
e) Потърсете обяснение на резултата от d, като използвате формулата на Хорнер.

f) Какво число ще се получи, ако в края на двоичното представяне на m добавим две нули?

11. ДИСКРЕТНО ПРЕДСТАВЯНЕ НА ИНФОРМАЦИЯТА

Навярно всеки е виждал как се бродират гоблени. Върху канавата – бод след бод – се появяват цветни кръстчета, които в своето съчетание пресъздават оригинала. Като прототип се използва копие от някоя картина, предварително разделено на малки квадратчета (пиксели) – по едно за всяко кръстче. Това е начинът един образ, сложен като рисунък и цветна гама, да бъде пренесен върху плата и да бъде наподобен по форма и цвят.





Такова раздробяване на образа на малки елементи се нарича **дискретизация**.

Колкото по-малки са кръстчетата – толкова по-качествен ще стане гобленът, но и трудът за ушиването му също ще нарастне.

Подобен подход за представяне и пренасяне на графична, звукова и други видове информация се използва широко в много технически системи. По подобен начин се предават днес телевизионни образи, факсови съобщения. Така се осъществяват мобилните комуникации и все по-голяма част от телефонните разговори. На подобен принцип се основават и най-качествените съвременни методи за аудио и видео записи, използващи така наречените компактни дискове.

От математиката е известно, че функцията y е дефинирана само за такива стойности на аргумента x, за които подкоренната величина е неотрицателна. С други думи казано, функцията y може да се изчислява само за стойности на аргумента x, за които $x-2 \geq 0$.

Алгоритъм А2:

1. Въведи стойност на аргумента x.
2. Изчисли стойността на израза $x - 2$.
3. Ако $x-2 \geq 0$, изпълни 4, иначе 1.
4. Изчисли стойността на функцията $y = \sqrt{x-2}$.
5. Изведи стойността на функцията y.
6. Край.

В този пример забелязваме, че инструкцията с номер 3 служи за насочване (разклоняване) хода на алгоритъма А2 в едната от две възможни посоки. Разклоняването се осъществява чрез условието: „Ако $x-2 \geq 0$". За едни стойности на аргумента x това условие е изпълнено (т. е. вярно), а за други – не. За тези стойности, за които е изпълнено условието, ходът на алгоритъма ще продължи с инструкцията 4, а за тези, за които не е изпълнено – с инструкция 1.

Ходът на А2, а и ходът на повечето други алгоритми, се управлява и разклонява в зависимост от някои междинно получени резултати.

Определение: Елементарно действие, което служи за разклоняване на алгоритъма в две посоки, се нарича **логическо действие** (или **логическа инструкция**).

Всяка логическа инструкция съдържа две възможни алтернативни продължения и условие за избор на едното от тях. Изпълнението на логическата инструкция започва с проверка на условието. Ако се окаже, че условието е вярно (т. е. изпълнено), се изпълнява едното елементарно действие, а ако условието се окаже невярно – другото (алтернативното). Всъщност верността на условието управлява хода (разклоняването) на алгоритъма.

Условия, които се използват в логическите действия за разклоняване на алгоритъма, се наричат **логически условия**. Тук ще подчертаем, че трябва да използваме само логически условия, верността на които може ефективно и еднозначно да бъде определена при всяко изпълнение на логическото действие.

Алгоритъм, който съдържа поне една логическа инструкция, се нарича **разклонен алгоритъм**. Алгоритъмът А2 е разклонен алгоритъм, защото съдържа логическа инструкция – елементарното действие 3.

Алгоритмите се прекратяват (завършват) при изпълнение на инструкцията „край". Следователно всеки алгоритъм трябва да доведе до решаване на поставената задача, преди да достигне инструкцията „край".

В Пример 1 и Пример 2 са представени два алгоритъма, означени съответно с А1 и А2. Описанието на алгоритмите е направено чрез текст, т. е. използван е естественият човешки (в случая – български) език. Последователността за изпълнение на елементарните действия се определя с помощта на поставената номерация.

Ето някои важни изисквания към алгоритмите:

– **Резултатност.** Това означава, че при каквито и начални условия да започне изпълнението на алгоритъма, винаги достига до адекватно решение на проблема, заради който този алгоритъм е бил създаден.

– Алгоритмите са „**крайни**", т. е. всеки алгоритъм достига до решение на съответната задача, след извършване на **краен брой** елементарни действия.

– **Детерминираност**. Това означава, че всяко изпълнение на алгоритъма при едни и същи начални условия трябва задължително да доведе до един и същ краен резултат.

ВЪПРОСИ И ЗАДАЧИ

1. Из завещанието на стар пират:

„... ще стигнете до високо дърво. Застанете до него и се обърнете на север. Направете 500 крачки напред и се обърнете на дясно на 60°. Като направите това шест пъти, ще се озовете на мястото, където е зарито съкровището."

На какво разстояние от дървото е зарито съкровището?

Ч. АЛГОРИТМИ И СТРУКТУРИ ОТ ДАННИ

Ч.1. АЛГОРИТМИ

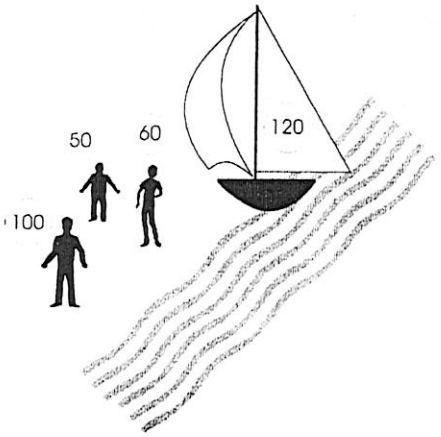
КАКВО Е АЛГОРИТЪМ?

Ежедневно хората се сблъскват с най-различни проблеми и задачи. Преодоляването на проблемите и решаването на поставените задачи човек постига благодарение на своите знания, опит и умения. Особено ценни за практическата дейност на хората са различните технологични знания – добре обмислени и проверени, описани под формата на рецепти, предписания, алгоритми, технологии, програми и т. н. Тези понятия се използват в различни области на обществената практика. Макар и наричани с различни имена, всички те имат обща същност – представляват описание на дейности и реда на тяхното изпълнение, което води до желани резултати. В областта на информатиката и информационното обслужване е приет и се използва терминът „алгоритъм".

Определение: Крайна последователност от елементарни действия (инструкции), изпълнението на които води до решаване на определен проблем, се нарича алгоритъм.

За всеки конкретен алгоритъм съществуват два субекта – „съставител" и „изпълнител". Алгоритъмът се измисля и описва от „съставителя" (обикновено човек) и е предназначен да бъде извършван от конкретен, отнапред известен „изпълнител" (човек или машина). **Елементарно действие** наричаме такова действие, което „изпълнителят" може да извърши, без да получава допълнителни указания или помощ.

Създаването и описването на алгоритъма трябва задължително да е съобразено с възможностите на потенциалния „изпълнител„. Следователно алгоритъмът може да включва само елементарни за „изпълнителя" действия, при това записани на разбираем за него език.



Пример 1

Баща и двамата му синове тръгнали на пътешествие, но неочаквано се озовали пред пълноводна река. Намерили малка лодка, която може да превозва не повече от 120 кг товар. Как да постъпят, за да преминат реката, ако бащата тежи 100 кг, а синовете му съответно по 50 и 60 кг?

Бащата и двамата му синове ще могат да преодолеят реката, като изпълнят алгоритъма А1:

А1

1. Двамата сина преминават на отсрещния бряг.
2. Единия от тях (без значение кой) връща лодката.
3. Бащата преминава реката сам.
4. Синът, който е на отсрещния бряг връща лодката.
5. Двамата синове окончателно преминават реката.
6. Целта е постигната. Край.

Алгоритъмът А1 може да бъде използван за преодоляване на реката, ако всеки от тримата умее сам да управлява лодката. Следователно елементарното действие в случая е управлението на лодката.

Елементарните действия, съставящи алгоритъма А1, се извършват последователно едно след друго. Такива алгоритми се наричат **линейни алгоритми**.

Пример 2

Съставете и опишете алгоритъм за пресмятане и извеждане на стойността на функцията $y = \sqrt{x-2}$.

2. Стълб е висок 20 м. Мравка се изкачва по него, като на ден изминава 5 м нагоре, а нощем слиза по 4 м надолу. Определете след колко дни мравката ще достигне върха на стълба.

3. Игра (играят двама души).

Има 15 предмета. Съперниците, редувайки се, могат да вземат 1, 2 или 3 предмета. Съставете и опишете алгоритъм на печеливша стратегия за играча, който прави първото вземане.

4. Съставете и опишете алгоритъм за пресмятане на стойността на функцията $y = f(x)$, дефинирана по следния начин:

а) $y = \dfrac{1}{x-6}$

б) $y = \dfrac{x}{(1+x)(x-9)}$

в) $y = \sqrt{x^2 - 3x + 4}$

г) $y = \begin{cases} x^2 - 1 & \text{за } x < 1 \\ 5x & \text{за } x \geq 1 \end{cases}$

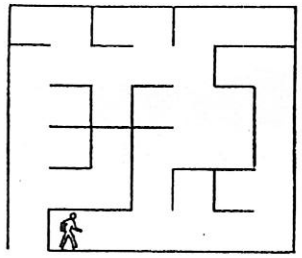
д)* $y = \begin{cases} x(1-x^2) & \text{за } x \leq 0 \\ 3x & \text{за } 0 < x < 1 \\ 2x+1 & \text{за } x \geq 1 \end{cases}$

5. Имате 9 монети, едната от които е фалшива (тя е по-лека). Разполагате и с везна. Съставете и опишете алгоритъм, с помощта на който може да бъде открита фалшивата монета с най-много две претегляния.

6. От 17 монети една е по-лека. Разполагате и с везна. Съставете и опишете алгоритъм, с помощта на който може да бъде открита фалшивата монета с най-много три претегляния.

7. От 12 футболни топки една е с нестандартно тегло. Разполагате и с везна. Съставете и опишете алгоритъм, с помощта на който може да бъде открита нестандартната топка с най-много четири претегляния.

9. Опишете алгоритъм, който трябва да следва човечето, за да излезе от лабиринта (правоъгълник с дължини на страните 7 m и 6 m).



Ч.2 НАЧИНИ ЗА ОПИСВАНЕ НА АЛГОРИТМИТЕ

Важен етап при създаването на алгоритъма е формалният му запис и документиране. За да може да бъде използван, алгоритъмът се описва от „съставителя" по подходящ (разбираем) за „изпълнителя" начин. Съществуват различни начини, с помощта на които се описва последователност от елементарни действия – с естествен език (словесно или текст), с фигури, блок-схеми, алгоритмични езици, машинни (компютърни) езици, езици за програмиране и т. н. Изборът на средството за описание се определя от характера, сложността и предназначението на алгоритъма и най-вече от възможностите на „изпълнителя".

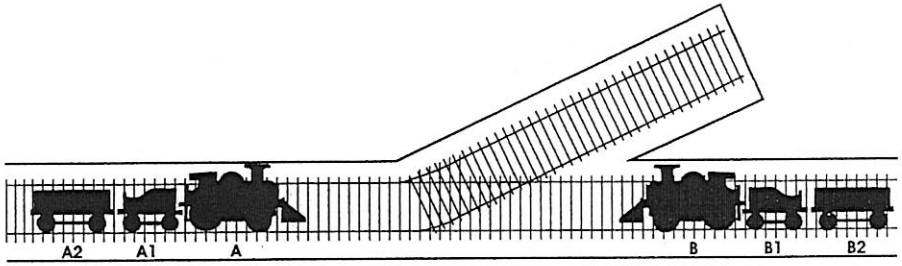
Тук ще бъдат обосновани, представени и илюстрирани най-често използваните в практиката начини и средства за описание (представяне) на алгоритмите.

ИЗРАЗЯВАНЕ НА АЛГОРИТМИ ЧРЕЗ ПОСЛЕДОВАТЕЛНОСТ ОТ ФИГУРИ

В предходния урок представяхме разгледаните алгоритми чрез текст, написан на български език. Ще отбележим, че използването на естествени езици е възможно (и то не винаги), когато „изпълнителят" е човек. Използването на естествен език за описание на по-сложни алгоритми е неудобно или невъзможно дори и за човека поради неточностите, двусмислията и липсата на подходящи конструкции в говоримите езици. За да се убедите сами в това, опитайте се да обясните словесно алгоритъм за разминаване на влаковете от следния пример:

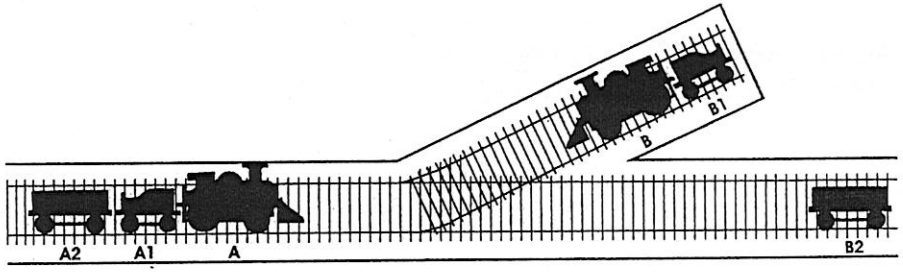
Пример 1

Два влака, всеки един от които е композиран от локомотив и два вагона, се срещнали на жп линия, която има глухо отклонение (фиг.1). Какви маневри трябва да направят машинистите, за да се разминат влаковете, ако глухото отклонение може да събере най-много вагон и локомотив?

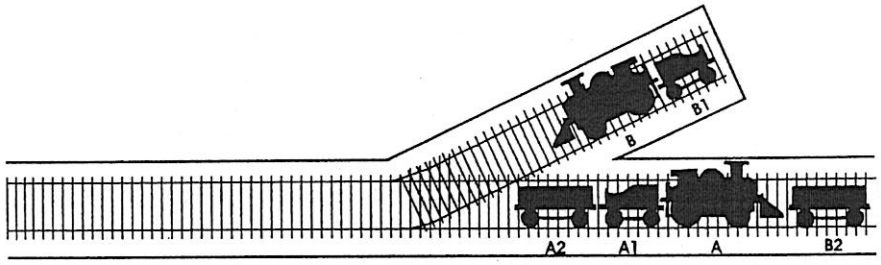


Фиг. 1

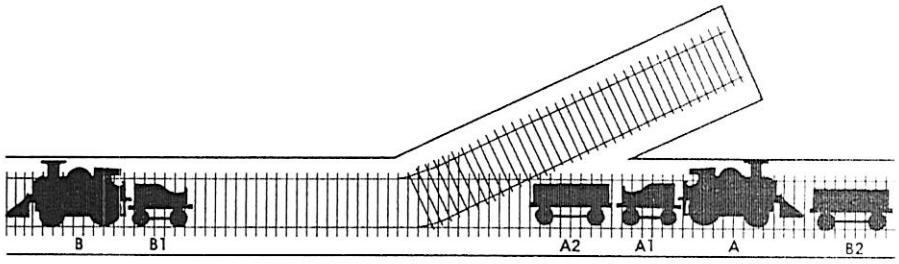
Опитайте се да обясните словесно последователността от маневри, които трябва да се извършат, за да се разминат влаковете. Словесният начин за описване решението на тази задача вероятно ще се окаже труден и дори неудачен. В този случай най-подходящо е алгоритъмът да се представи чрез последователност от фигури, така, както е направено на фиг. 2–7. Те илюстрират маневрите, които трябва да извършат железничарите, за да разминат двата влака.



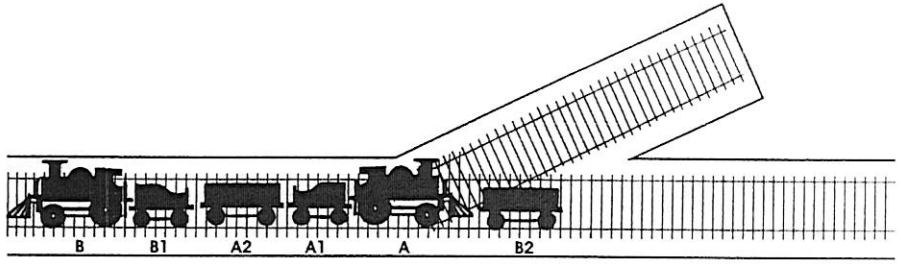
Фиг. 2



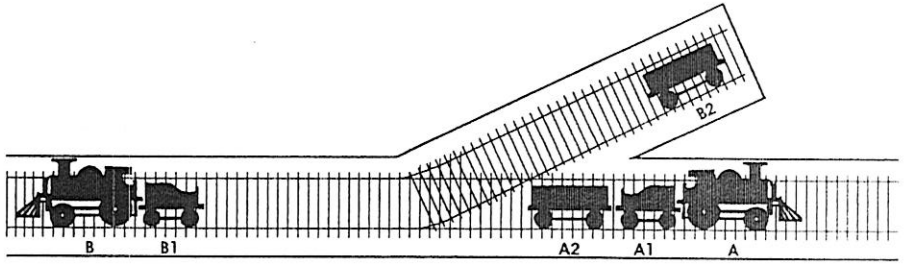
Фиг. 3



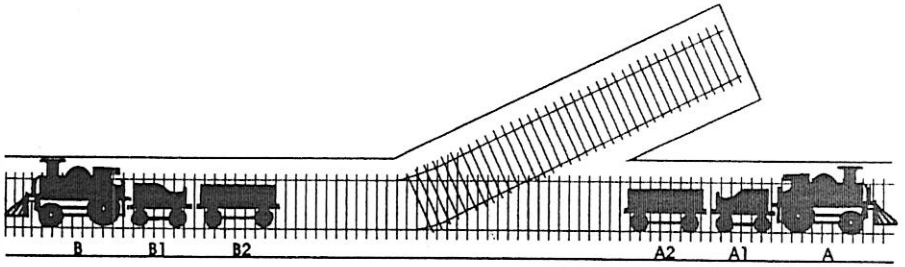
Фиг. 4



Фиг. 5



Фиг. 6



Фиг. 7

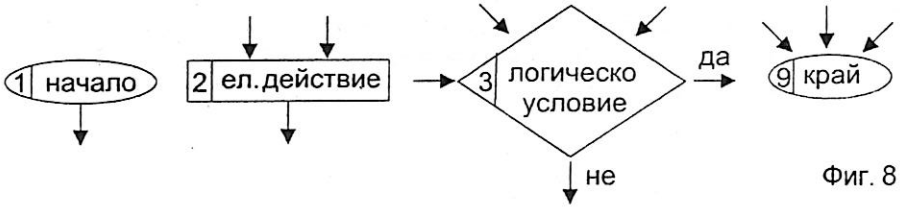
3. Логически блок

Блоковете с такава форма се използват за изразяване на логически действия. Във вътрешността на блока се записва логическото условие. Изпълнението на блока започва с проверка на логическото условие. Ако логическото условие се окаже изпълнено (т. е. вярно), се изпълнява последователността от дейности, указана със стрелката „да". Когато логическото условие е невярно, се продължава в посока „не". Всъщност логическият блок се използва за разклоняване на алгоритъма. За всеки логически блок в блок–схемата съществува поне един предходен и точно два следващи го блока.

4. Блок за край на алгоритъма

С такъв блок завършва всяка блок–схема. За него съществува поне един предходен и няма следващ блок.

Както вече знаем, последователността на изпълнение на блоковете (т. е. на елементарните действия) в блок–схемата се определя от стрелки. Също така блоковете могат да се номерират (по показания на фиг. 8 начин) и с помощта на поставената номерация да бъдат цитирани (сочени) при осъществяване на преход.



Фиг. 8

Описването и документирането на алгоритмите с помощта на блок–схема е прието да се нарича **блок–схемен език**. Блок–схемният език е създаден в най-ранните години от развитието на информатиката. Основните му качества – простота, нагледност и лекота при проследяване последователността на елементарните действия

(т. е. на логиката на алгоритъма), го прави приложим и използван в редица ситуации и в наши дни.

Задача 1

Съставете и опишете с блок–схема алгоритъм, който:

а) въвежда конкретни числови стойности на параметрите a и b;

б) разменя стойностите на параметрите a и b;

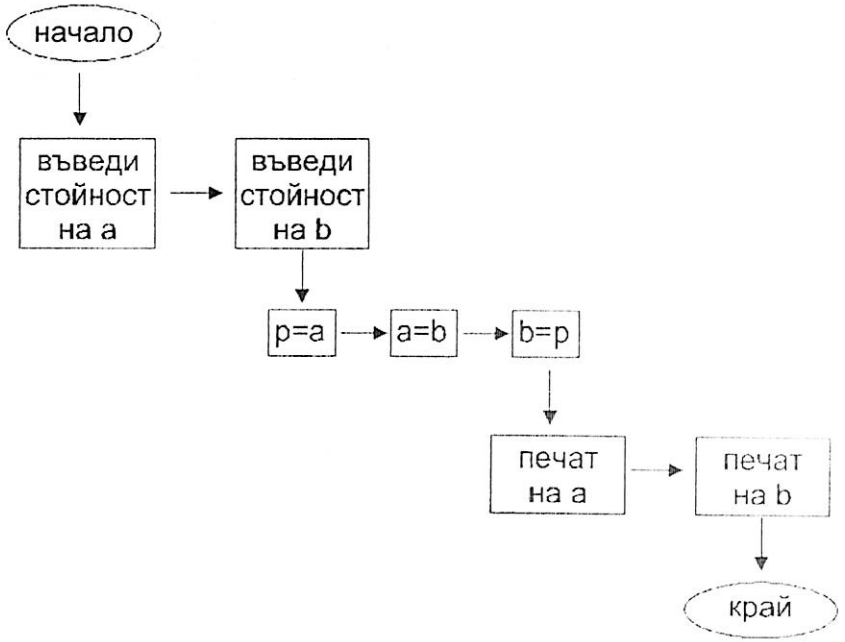
в) извежда (отпечатва) новите стойности на a и b.

За да се досетите за алгоритъма, помислете върху решението на следната сходна, но „по-лесна" практическа задача:

Едната от две чаши съдържа вода, а другата – лимонада. Разменете съдържанието на чашите.

За да се извърши размяната на течностите ... е необходима трета чаша.

Аналогично, за да разменим стойностите на параметрите a и b, е необходимо да използваме трети (допълнителен) параметър например p.



Изразяването на алгоритми чрез последователност от фигури се използва често в ежедневната практика, защото:

1. Този начин за представяне на алгоритми е езиково неангажиран, т. е. разбираем е и може да се използва от хората независимо от езиците, с които говоримо или писмено си служат.

2. Дейностите и последователността им са добре онагледени, което създава условия да бъдат разбрани и извършени от широк кръг потребители с относително ниска квалификация.

ПРЕДСТАВЯНЕ НА АЛГОРИТМИТЕ ЧРЕЗ БЛОК–СХЕМИ

Много често алгоритмите са твърде сложни и представянето им чрез фигури е невъзможно. Изразени пък словесно или с текст на естествен език губят редица свои важни качества – яснота, прегледност, точност, а дори и еднозначност.

В редица случаи е удобно, нагледно и удачно елементарните действия да бъдат описани с помощта на възприета символика и записани в геометрични блокове. Отделните блокове се свързват със стрелки, като стрелките определят последователността на изпълнението им. Подобно средство за описание и документиране на алгоритми се нарича **блок–схема**.

При създаване на блок–схемите обикновено си служим с ограничен набор от различни по форма блокове. В блоковете с еднаква форма е прието да се записват еднотипни по своята същност елементарни действия.

В блок–схемите ще използваме следните означения:

1. Блок за начало на алгоритъма

С такъв блок започва всяка блок–схема (т. е., описанието на алгоритъма). Началният блок няма предходен блок, а има точно един блок, който го следва. Следователно към него не сочи стрелка, а от него излиза точно една стрелка.

2. Блок за елементарни действия

В блоковете с тази форма се записват елементарните действия. За всеки блок от този тип в блок-схемата съществува поне един предходен и точно един следващ го блок.

Елементарните действия, свързани с изчисления, се задават с помощта на аритметични изрази (формули), в които участват числови константи и/или параметри (аргументи). Аритметичните изрази (формулите) се записват чрез символиката и по правилата, определени в математиката. Елементарните действия за изчисления се задават в блок със следната най-обща структура:

Извършват се в два последователни етапа:

1. „Изпълнителят" пресмята стойността на аритметичния израз, записан в дясната страна на знака „=". Изчисленията се извършват с текущите стойности на участващите в него аргументи, при спазване приоритета на аритметичните операции.

2. Полученият резултат се присвоява и запомня като стойност на записания отляво параметър. В този смисъл знакът „=" представя операция за присвояване. Например при изпълнение на последователността от блокове:

а) → x=22+5 → изведи стойността на x →

се извежда числото 27.

б) → x=10 → x=20 → y=x-5 → изведи стойността на y →

ще се изведе числото 15 (а не 5), защото стойността на y е изчислена с текущата (т. е. последно присвоената) стойност на параметъра x. Ще напомним, че параметърът x „помни" само последната стойност, която му е присвоена.

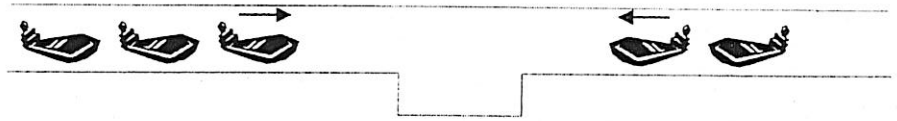
в) → x=1 → x=x+2 → изведи стойността на x →

извежда числото 3. Обяснете защо?

г) → p=2 → z=3(5p-1) → p=1 → изведи стойността на z → изведи стойността на p

извежда последователно числата 27 и 1, защото промяната на стойността на променливата p, не води до промяна на вече изчислената стойност на параметъра z.

3. На дадената по-долу фигура е изобразен плавателен канал, по който плуват едни срещу други два конвоя от кораби. Каналът има разширение, в което може да се побере само един кораб. Съставете и опишете чрез последователност от фигури маневрите, които трябва да извършат капитаните, за да се разминат корабите.



4. Съставете и опишете алгоритъм за пресмятане стойността на функцията $y = f(x)$, дефинирана по следния начин:

а) $y = \dfrac{1}{2-x}$

б) $y = \dfrac{1+x}{x} - \dfrac{1-x}{x+9}$

в) $y = \sqrt[4]{3-4x+x^2}$

г) $y = \begin{cases} 3-x^3 & \text{за } x < -3 \\ 9-7x & \text{за } x \geq -3 \end{cases}$

д) $y = \begin{cases} x^3(x+1) & \text{за } x < 0 \\ -2x & \text{за } 0 \leq x < 1 \\ 3x-5 & \text{за } x \geq 1 \end{cases}$

5. Опишете с блок–схема алгоритмите за решаване на задачите от предходния урок с номера: а) 6; б) 7; в) 8.

6. Съставете и опишете с блок–схема алгоритъм, които:

а) въвежда дължините на три отсечки;

б) проверява и съобщава дали с тези три отсечки може да се построи триъгълник.

7*. Опишете с блок–схема алгоритъма за решаване на линейното уравнение от вида

$ax = b$,

където a и b са коефициенти, а x е неизвестно.

9. ЕЗИК И СРЕДА ЗА ПРОГРАМИРАНЕ

Вече знаем, че веднъж създаден (измислен) алгоритъмът за решаване на дадена задача може да бъде записан чрез различни средства – текст на естествен език, последователност от фигури, блок–схема. Използването на всяко едно от тези средства има своите предимства, но и недостатъци, което го прави удобно и приложимо в ограничен кръг ситуации. Изборът на средството зависи от сложността и предназначението на алгоритъма и най-вече от възможностите на „изпълнителя“. Както вече отбелязахме, представените дотук средства за описание на алгоритми са приложими единствено и само когато „изпълнителят“ е човек, а не машина.

Желанието на хората да автоматизират изпълнението на информационните дейности премина през различни етапи, но в крайна сметка доведе до създаване на машина, която може да „разбере“, „запомни“ и изпълни сложни алгоритми, чрез които да обработва данни и/или управлява сложни технологични процеси. Тази машина се нарича компютър и на съвременния етап се използва като неуморим и високоскоростен „изпълнител” на написани от човека алгоритми.

За да може човекът да „обясни“, а компютърът да „разбере“ и изпълни съставения алгоритъм, е необходимо да използват (разбират) общ език, на който да комуникират. За да изясним и вникнем в същността на проблема, ще разгледаме една проста ежедневна ситуация:

Срещат се българин и немец. Как могат да общуват?

Не е трудно да си представим възможните алтернативи:

1. Българинът да научи немски и да разговарят на немски.

2. Немецът да научи български и разговорът да се проведе на български.

3. И двамата да научат „трети“ език (например английски), чрез който да се разберат.

Аналогично очакваме „диалогът“ между човека и компютъра да се реализира по някои от следните три различни начина:

Начин 1

Компютърът да „научи“ и „разбира“ (поне писмено) естествен човешки език. Засега това технологично е невъзможно и все още

Задача 2

Уравнение от вида

$ax^2 + bx + c = 0 \quad (a = 0)$

се нарича квадратно уравнение. Корените на това уравнение (ако съществуват) се изчисляват по формулите:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Съставете и опишете с блок–схема на алгоритъм, който решава квадратни уравнения.

Едно възможно решение на поставената задача е представено на фиг. 9.

Фиг. 9

У$_1$. Кое важно свойство ще загуби алгоритъмът от фиг. 9, ако в него се пропусне елементарното действие, означено с 1?

У$_2$. В урока са поставени две задачи. Посочете коя от тях е решена с линеен и коя с разклонен алгоритъм.

ВЪПРОСИ И ЗАДАЧИ

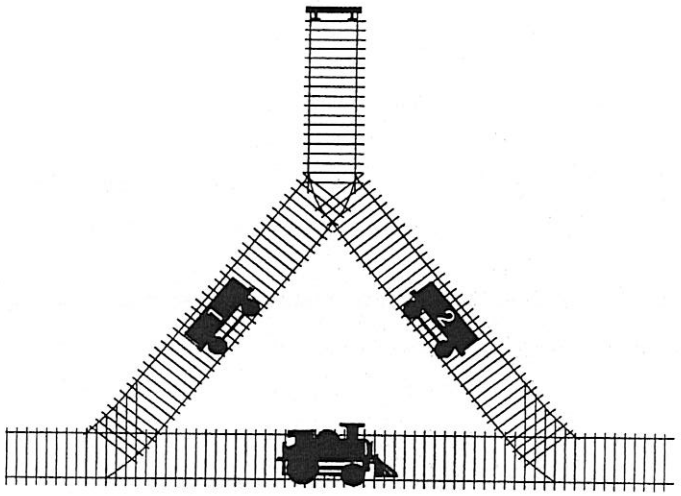
1. Посочете примери от вашето ежедневие, в които са използвани:

а) последователност от фигури;

б) блок–схеми;

за описване на последователност от дейности.

2. Дадената по-долу фигура представлява схема на главна ж.п. линия с две отклонения. Във всяко отклонение има по един вагон. В глухата линия може да влезе само един вагон или локомотивът.



Изразете чрез последователност от фигури маневрите, които трябва да извърши машинистът, за да размести вагоните, а локомотивът да се върне на старото (изходното) място:

а) насочен в същата посока;

б) обърнат в противоположна посока.

е в сферата на фантастиката. Разбира се, подобни опити се правят и са налице първите обнадеждаващи резултати.

Начин 2

Човекът да научи „езика" на компютъра. Както знаем, всеки компютър има свой собствен език – езика на централния му процесор. Този език се нарича **машинен** и съдържа „думи", състоящи се (написани) само от 0 и 1. Тези думи могат да бъдат машинни команди (инструкции), двоични константи или адреси в оперативната памет. Всеки централен процесор разполага със собствен набор от около стотина команди, които разбира и които може да изпълнява. Алгоритъмът се записва чрез последователност от тези команди и полученият „текст" представлява **програма на машинен език**.

Различните компютри боравят с различна по дължина машинна дума – 4, 8, 16, 32, 64 бита. По тази характеристика се различават 4-битови, 8-битови, 16-битови, 32-битове, 64-битови компютри, а в бъдеще вероятно ще има и компютри с по-големи възможности.

Ще разгледаме един хипотетичен 8-битов централен процесор, който:

– разполага с един специален блок (да го означим с А и наречем акумулатор) за временно съхраняване на данни или междинни резултати;

– може да изпълнява описания по-долу набор от машинни команди.

00000001 $C1_{(2)}C2_{(2)}$

Тази команда означава: „Събери стойността на двоично записаната константа С1 със стойността на двоичната константа С2 и резултата запази в акумулатора А".

00000100

„Отпечатай съдържанието на акумулатора А".

00000000

„Край на машинната програма".

Знаем също, че 11 е двоичното представяне на числото 3, а 101 е двоичното представяне на числото 5. Тогава последователността от 8-битови думи:

00000001 00000011 00000101 00000100 00000000

представлява машинната програма, която събира числата 3 и 5 и извежда получения резултат.

Програмирането на машинен език е първият етап в еволюцията на общуването между човека и компютъра, но използването на машинен език е трудно достъпно (разбираемо) за хората и е по силите на силно ограничен кръг специалисти (системни програмисти). Ето някои очевидни факти, които силно затрудняват използването на машинния език:

– програмистът трябва да познава много добре централния процесор и цялостната архитектура на компютъра, за които е предназначена програмата;

– програмата, написана на машинен език, представлява дълга последователност (низ) от 0 и 1, което я прави трудна за разчитане и още по-трудна за разбиране;

– програмистите трябва да знаят и помнят начина на записване (синтаксиса) и тълкуването (семантиката) на стотина команди, записани кодирано с двоични числа;

– програмистът сам планира разполагането на данните и програмата в оперативната памет на компютъра, като трябва да помни адреса на всяка данна или междинно получен резултат;

– програмата може да работи само на компютър с централен процесор, за който е създадена;

– машинните програми се въвеждат трудно и бавно, като неминуемо се допускат грешки;

– откриването и поправянето на допуснатите в програмата грешки е изключително сложен и трудоемък процес.

Начин 3

Да бъде създаден език за програмиране.

В исторически план първата стъпка в тази посока е било създаването и използването на т. нар. **асемблерни езици**. Тези езици представляват естествено развитие на машинните езици, в които:

– кодовете на командите са заменени със съответни думи на естествен език;

– използват се имена (вместо адреси) за означаване на полета с данни;

– константите могат да бъдат записвани в двоична, десетична или шестнадесетична бройна система.

В този смисъл, ако заменим кодовете на машинни команди в горния пример със съответните им думи на английски език:

00000001 – ADD (събери)
00000100 – OUTPUT (изведи)
00000000 – END (край)

ще получим най-обща представа за асемблерна програма:

```
ADD 3, 5, A
OUTPUT  A
END
```

Необходимо е да отбележим, че асемблерната програма (за разлика от машинната) не може директно да бъде изпълнена дори и на компютъра, за който е създадена. За да бъде изпълнена програма на асемблер, трябва първо да бъде „преведена" на съответния машинен език, след което преведеният (машинният) вариант – подаден на компютъра за изпълнение. За да бъде автоматизиран процесът на „превод" са били създадени специални програми, наречени **транслатори**.

През годините бяха разработени и намираха приложение не малко системи, съдържащи:

– асемблерен език;

– текстов редактор (програма), с който се въвежда и коригира текстът (кодът) на асемблерната програма;

– транслатор за превод от този език на конкретен машинен език.

Това са по същество първите прости **системи за програмиране**.

Използването на асемблерни езици облекчи до известна степен процеса на програмиране, направи програмата по–четима, но реално доближи човека към компютъра, а не компютъра към човека. Ето защо асемблерните езици постепенно загубиха своята роля в масовото програмиране и в наши дни се използват само за създаване на отделни фрагменти в съвременните сложни приложни програми.

ЕЗИК ЗА ПРОГРАМИРАНЕ

Затрудненията, свързани с използване на машинните и асемблерните езици, пораждат идеята за създаване на „езици от високо ниво". В тях елементарните действия се описват с ограничен брой думи или словосъчетания от естествен (най-често английски) език, а изчисленията – по начин, близък до използвания в математиката.

Съществена практическа стъпка за облекчаване описанието на числовите пресмятания е направена от сътрудници на фирмата IBM, ръководени от Джон Бекъс. През 1954 г. те създават езика Fortran. Наименованието на езика е получено от съкращения в английския израз FORmula TRANslation, който преведен на български означава – превод на формули. Самото наименование подсказва новото качество, което притежава този език. Пресмятанията във Fortran се записват директно с изрази (формули) близки по вид и смисъл на тези в математиката. Новият начин за представяне на формулите в програмата и пресмятане на техните стойности е позволил на колектива на Бекъс да създаде програма транслатор, която автоматично превежда програмата от Fortran на машинен език. Противно на тогавашните очаквания, получената автоматично машинна програма се оказва достатъчно добра и това поставя началото на нов етап във взаимодействието на човека с компютъра.

Започва създаването, развитието и използването на т. нар. **езици за програмиране**. При описанието на алгоритъма с език за програмиране се създава текст, наречен **програма**. Обикновено текстът на програмата включва символично описание на данни и последователност от действия (алгоритъм), с които ще се обработват тези данни. Данните се описват като константи и/или променливи от определен тип. Действията се означават с ограничен брой думи или прости словосъчетания (най-често от английски език), които се наричат **оператори**.

ТРАНСЛАТОРИ

Програмата, написана на език за програмиране, съдържа оператори, различни от командите на машинните езици и следователно не може да бъде директно „разбрана" и изпълнена от компютър. Ето защо се налага най-напред да се извърши „превод" на програмата.

Преводът се извършва автоматично от друга, предварително създадена и въведена в компютъра програма, наречена **транслатор**. В резултат от автоматичния превод (т. е. от транслацията) се получава преведена (изпълнима) компютърна програма, еквивалентна на първоначалната.

Използват се два различни подхода (интерпретивен и компилативен) за превод на програмите. Транслаторите, които ги реализират, се наричат съответно **интерпретатори** и **компилатори**.

Повечето съвременни **интерпретатори** са интегрирани в цялостна система за програмиране. Интеграцията позволява на интерпретатора да извършва синтактична проверка и контрол на всеки един оператор още при въвеждането му в текста на програмата. На практика интерпретаторът „следи" програмиста и не му „разрешава" да допуска правописни грешки.

Интерпретативният подход представлява „схема за превод", аналогична на симултантния превод, използван за диалог между хора, говорещи различни езици. Интерпретаторът избира, „превежда" и изпълнява операторите от програмата един след друг в тяхната алгоритмична последователност. Всеки избран оператор се разпознава и проверява (анализира) дали е записан по правилата на езика. Ако записът се окаже правилен, интерпретаторът стартира своя (съдържаща се в него) машинна програма, която извършва съответното действие. След това се преминава към интерпретация на следващия според алгоритъма оператор и т. н., докато се достигне до оператора за край.

В процеса на интерпретация (изпълнение) на програмата се извършва и разполагането на променливите в оперативната памет на компютъра. За всяка променлива се заделя определено място в оперативната памет, размерът на което зависи от типа на променливата. Това място интерпретаторът фиксира при първата си среща с конкретната променлива, след което винаги там записва или оттам прочита текущата ѝ стойност.

Всяко изпълнение на програма изисква отново разпознаване и анализ, т. е. интерпретация на включените в нея оператори. Този факт обективно забавя изпълнението на алгоритъма. От друга страна интерпретаторите:

– са по-прости компютърни програми от компилаторите;

– притежават по-ефективни средства за откриване и диагностициране на грешки;

– могат да се изпълняват на персонални компютри с по-скромни технологични възможности.

Тези предимства правят интерпретаторите особено подходящи за начално обучение и програмиране от неспециалисти. Най-популярните езици, при които се използват интерпретатори, са различните версии на Basic и LOGO.

Компилаторите са транслатори, които „превеждат" (компилират) цялата програма от език за програмиране на машинен език. Компилацията, т. е. преводът, е сложен процес, който най-общо протича в няколко взаимно свързани етапа:

1. Синтактичен и семантичен анализ на текста на програмата. Извършва се проверка на правилното записване и употреба на операторите и конструкциите от езика, използвани в текста на програмата. При този етап компилаторът автоматично открива допуснатите „правописни" и конструктивни грешки в текста на програмата и ги съобщава на програмиста.

2. За всяка променлива, използвана в програмата, се определя дължината на полето (последователност от един или няколко байта), в което може да се разположи стойността ѝ. Дължината на полето зависи от типа на данните, които са описани с променливата. Най-често целочислените променливи се записват в 2 байта, а реалните – в 4 байта.

3. За всеки оператор се извлича (от служебна за езика библиотека) и настройва фрагмент от машинна програма, който е предназначен да реализира този оператор.

4. В крайна сметка компилаторът свързва машинните фрагменти и описанието на полетата за данни (променливите) в единна машинна програма. В процеса на свързване компилаторът вгражда допълнително един служебен фрагмент в преведената програма. Този фрагмент управлява хода на изпълнението на машинната програма при всяка нейна употреба, като:

– оптимално разполага отделните части на програмата в свободните пространства на оперативната памет;

– динамично резервира и освобождава полетата, в които се разполагат стойностите на променливите (данните);

– управлява последователността на изпълнение на отделните части в програмата, както и обмена на данни между тях.

Тестването на компилираната програма се извършва от програмистите. Потребителите получават и използват само изпълнимия (компилирания) вариант на готовата програма. Прието е името на изпълнимия вариант на програмата да притежава стандартното разширение .EXE. Например, ако текстът на компютърна програма, написана на езика Pascal, е записан във файл с име SORT.PAS, то името на изпълнимия (компилирания) вариант на тази програма ще бъде SORT.EXE.

Новото качество, достигнато в езиците за програмиране, е, че програмистът е освободен от досадната грижа сам да разполага данните и програмата в оперативната памет, да помни адреса на всяка данна и междинно получен резултат. В текста на програмата данните и междинните резултати се означават чрез **променливи**. Обработката на данните се описва символично (абстрактно) в програмата. Същинската обработка на данните се осъществява в процеса на изпълнение на машинната програма.

РАЗВИТИЕ НА ЕЗИЦИТЕ ЗА ПРОГРАМИРАНЕ

От създаването на Fortran до наши дни са създадени и развивани стотици езици за програмиране. Първоначално създадените езици са били проблемно ориентирани, т. е. съдържат подходящи средства за решаване на определен кръг задачи. Така например:

– Algol-60 е създаден през 1960 г. и първоначално е бил предназначен за числени пресмятания. Идеите, заложени в този език, са изиграли важно теоретично значение в по-нататъшното развитие на езиците за програмиране.

– Cobol е създаден през 1960-1961 г. и е предназначен за обработка на икономическа информация;

– Lisp е създаден през 1959 г. През последните години получават широко разпространение в областта на изкуствения интелект.

– Snobol. Предназначен е за обработка на низове (текстова информация). Създаден е през 1962-1963 г, и е прилаган за разработване на транслатори;

– Simula-67. Създаден е през 1967 г. и е ориентиран към задачи за симулационно програмиране;

– LOGO е създаден в края на 60-те години за нуждите на обучението;

– Basic. Първоначално е бил създаден за целите на обучението. Езикът получава изключително широко приложение при появата на микрокомпютрите. Масовото му използване в днешно време е свързано с влагането му като средство за програмиране в най-широко използваните приложни програми.

Успоредно с развитието на компютърната техника (бързодействие, памет и т. н.) се появява възможността за създаване на универсални езици, с помощта на които могат да се решават широк кръг от задачи. Последователно във времето са създадени и масово използвани следните езици за програмиране: PL/1 (1964 г.); Algol-68 (1968 г.); Pascal (1971 г.); C (1978), Ada (1980 г.). В последното десетилетие значително разпространение получиха така наречените „обектно-ориентирани" езици. Типични техни представители са C++ (1985 г.) и Java. Използват се от професионалните програмисти за създаване на сложни софтуерни продукти, много от които използват ресурсите на локална или глобална (Internet) компютърна мрежа.

СИСТЕМИ ЗА ПРОГРАМИРАНЕ

Съществуващите компютърни езици са твърде разнообразни и многобройни. Ще отбележим, че **не всички** са предназначени за описание на алгоритми. Такива са например езиците Prolog (създаден 1972 г.) и HTML. Prolog се използва за решаване на задачи в областта на изкуствения интелект, а езикът HTML е предназначен за създаване на уебстраници в Internet. И двата езика почти нямат възможности за описание на алгоритми.

Облекчаване труда на програмиста и увеличаване на неговата продуктивност се осъществява с използването на т. нар. **системи за програмиране**. Системите за програмиране представляват комплект от средства (програми), с помощта на които се създава компютърна програма: въвеждане, редактиране и съхраняване на текста на програмата; транслирането; тестването; документирането; изпълнението и т. н. Един минимален набор от средства, който съдържа системата за програмиране, е следният:

– **Език за програмиране;**

– **Текстов редактор.** Използва се за въвеждане, съхраняване и коригиране (редактиране) на текста на програмите.

– **Транслатор** (интерпретатор или компилатор) на език за програмиране.

– **Система за проверка на грешки в алгоритъма, наричана дебъгер**. Терминът произлиза от английската дума debug, означаваща процес за изчистване на грешки в компютърна програма. Чрез дебъгера програмистът може да проследи както последователността на изпълнение на операторите (алгоритъма), така и междинните стойности на някои променливи. Дебъгерите са ценно средство за откриване на грешки в логиката на алгоритъма, които транслаторите не са в състояние да локализират.

– **Система за настройка**. Позволява на програмиста да настройва начина на използване и функциите на отделните компоненти в средата за програмиране (включително и на транслатора).

– **Система за поддържане и използване на библиотеки с програми**. Дава възможност на програмиста да използва готови програмни модули (създадени от него, негови колеги или съдържащи се в самата система за програмиране) като градивни елементи при създаване на новите програми.

СРЕДИ ЗА ПРОГРАМИРАНЕ

В началото на деветдесетте години на миналия век производителите започнаха да включват в системите за програмиране и т. нар. управляващи програми. Управляващата програма интегрира (свързва в едно цяло) отделните компоненти на системата и осъществява цялостното взаимодействие с програмиста. Такива системи получиха названието **среди за програмиране**, а типични техни представители са: QBasic, Turbo Pascal (версии: 5.0, 5.5, 6.0, 7.0), Turbo C, Comenius Logo и др. Появиха се и среди за програмиране, които позволяват отделните модули на една програма да бъдат създавани с използването на различни езици за програмиране.

В края на 20-ти век лавинообразно нараства потреблението на информационни и комуникационни технологии. Това наложи софтуерната индустрия да търси и създаде нови „инструменти", които да повишат съществено производителността на програмиста. В тази връзка получиха развитие две идеи.

Едната от тях доведе до развитие и масово използване на системи, които опростяват и автоматизират процеса на структуриране, въвеждане, поддържане и използване на данни. Такива софтуерни „инструменти" получиха названието „системи за управление на бази от данни". Възможностите на подобни системи са пряко свързани с

потенциала на компютъра и операционната система, за която са предназначени.

Втората идея е свързана със създаването на среди за „визуално" програмиране. Характерното за тези среди е, че съдържат софтуерни инструменти („построители"), с помощта на които се автоматизира създаването на текста на програмата. Използването на „визуално програмиране" промени коренно технологията за производство на софтуер. Програмистът не се грижи пряко за цялостното описание на алгоритъма. Средите за визуално програмиране предлагат система от визуални обекти, всеки от които притежава изглед, елементи и свойства, които могат да бъдат определяни (задавани). Програмистът избира и разполага желаните обекти върху система от взаимно свързани екрани (форми), след което настройва реакцията на всеки обект (промяна на формата и елементите върху тях) при възникване на определен кръг събития в хода на изпълнение на програмата. Системата „построител" автоматизира в голяма степен създаването на текста на програма, написана на конкретен език за програмиране. Подобни технологии за създаване на софтуерни продукти е прието да се наричат „програмиране, предизвикано от събития в среда на графичен потребителски интерфейс". Най-разпространени представители на среди за визуално програмиране са:

– Visual Basic с език за програмиране и транслатор на Basic;
– Delphi с език за програмиране и транслатор на Pascal;
– Visual C, съставящ програма на езика C;
– Java.

ВЪПРОСИ И ЗАДАЧИ

1. Какви предимства дава използването на асемблерен език в сравнение с:

а) програмирането на машинен език;

б) използването на език за програмиране от високо ниво?

2. Обяснете същността на човешко-машинния „диалог", чрез които алгоритмите се зададат и изпълняват на компютър.

3. Възможно ли е да съществува интерпретатор и компилатор за един и същ език за програмиране?

4. Защо средите за програмиране, използващи интерпретатор, не се използват от програмистите за създаване на приложни програми?

4.4. ОПИСВАНЕ И РЕАЛИЗАЦИЯ НА АЛГОРИТМИТЕ ЧРЕЗ КОМПЮТЪРНИ ПРОГРАМИ

До настоящия момент съставихме и описахме множество алгоритми. Описанието правихме с естествен (български) език, последователност от фигури или чрез блок–схеми. Така изразени алгоритмите могат да бъдат разбрани и изпълнени само от човек, но не и от машина. Желанието да автоматизираме изпълнението на алгоритмите налага да използваме компютър в ролята на „изпълнител". За да използваме компютър като универсален „изпълнител" на алгоритми, трябва да знаем конкретен език и да разполагаме със среда за програмиране. В по-нататъшното изложение ще въведем и използваме ограничен кръг команди, данни (числови, текстови) и правила за програмиране в езика QBasic, които са достатъчни, за да изпълним изучаваните алгоритми.

Програмата на QBasic се състои от редове. Всеки ред съдържа един или няколко оператора, разделени с двоеточие. Редовете могат (ако е необходимо) да бъдат отбелязвани (т. е. да започнат) с етикет. Етикетът е последователност от символи (букви и/или цифри от английската азбука), завършваща с двоеточие. Когато етикетът е естествено число, двоеточието може да не се запише.

Реалните числа се представят приближено с десетични дроби. Цялата и дробната част на десетичните дроби се разделят със символа „точка" (.). Текстовите константи се ограждат в кавички („).

Имената на променливите са последователност от букви и цифри, започваща с буква. В по-нататъшното изложение ще използваме само числови и текстови променливи, въпреки че QBasic притежава средства за обработка на всички стандартно използвани в езиците за програмиране данни. Стойностите на числовите променливи могат да бъдат реални числа, а на текстовите променливи – крайна последователност (до 256) от символи.

Програмата се въвежда и коригира (редактира) с помощта на текстовия редактор в средата за програмиране. Текста на програмата обикновено се записва (съхранява) на харддиска (или на дискета). Първото записване се извършва с последователността от команди на средата за програмиране File —> Save As (запази като), при което трябва да въведем името на файла и посочим пътя до папката

за запис. При първото записване системата „запомня" въведеното пълно име и следващи записи могат да се правят опростено – File —> Save.

Последователността от команди Run —> Start стартира въведената програма.

Последователността от команди File —> New дава възможност да пишем нова програма.

За да "засричаме" на QBasic, ще въведем и използваме няколко оператора:

ОПЕРАТОРИ CLS И END

CLS представлява команда за изчистване екрана на монитора. Наименованието на командата произтича от съкращение на английския израз „Clear Screen", който на български означава „Изчисти Екрана" . С този оператор ще започват нашите програми, а ще завършват с оператора END (END на български означава „КРАЙ").

ОПЕРАТОР ЗА ПРИСВОЯВАНЕ СТОЙНОСТ НА ПРОМЕНЛИВА

Общият вид на този оператор е:

име на променлива = аритметичен израз

В аритметичните изрази се използват обичайните за математиката действия, записани както следва: събиране (+), изваждане (-), умножение (*), деление (/) и степенуване (^). Изчисляването на израза се извършва при спазване на познатия от математиката приоритет:

1. Изчисляват се изразите в скобите (ако има такива).
2. Степенуване.
3. Умножение и деление.
4. Събиране и изваждане.

Ако са записани няколко последователни действия с еднакъв приоритет, то те се извършват последователно от ляво на дясно. Например при изпълнение на командата за присвояване:

b = 12 * 4 / 6

най-напред 12 ще бъде умножено с 4, а полученият резултат (48) – разделен на 6. В крайна сметка на променливата b ще бъде присвоена стойността 8.

Изпълнението на оператора за присвояване се осъществява в два последователни етапа:

1. Най-напред се пресмята стойността на аритметичния израз, записан в дясната страна на знака „=". Изчисленията се извършват с текущите стойности на променливите, участващи в него, при спазване приоритета на аритметичните операции.

2. Полученият резултат се присвоява (запомня) като стойност на променливата, записана отляво на знака за присвояване „=".

Задача 1

Съставете програма, която пресмята лицето на трапец с дължини на основите съответно: 10.5 и 6.66 см и височина 5 см.

Както знаем от математиката, формулата за пресмятане на лицето на трапеца е $s = \frac{a+b}{2}h$, където с a и b са означени основите, а с h – височината на трапеца.

Тогава последователността от оператори:

```
CLS
   a = 10.5
   b = 6.66
   h = 5
   s = (a  + b) / 2 * h
END
```

Програма 1

представлява програма, която компютърът трябва да изпълни, за да изчисли желаното лице.

При изпълнение на програмата на променлива a ще бъде присвоена стойността 10.5, на b – стойността 6.66, а на h – стойността 5. QBasic ще изчисли израза (a + b) / 2 * h, като извлече (от оперативната памет) и използва текущите стойности на променливите участващи в него. Получената (изчислената) стойност ще бъде присвоена на променливата s.

Съставената по-горе програма ще извърши пресмятанията, но има два съществени недостатъка:

– резултатът от изчисленията остава „скрит" за човека в оперативната памет на компютъра, защото е стойност на променлива s;

– не може да се използва за изчисляване на лицата на други трапеци, освен на този, с посочените конкретни размери.

Езиците за програмиране притежават средства, с помощта на които се преодоляват подобни недостатъци.

Всеки език за програмиране притежава поне един оператор, с помощта на които се указва на компютъра да покаже на монитора (или отпечата на хартия) стойностите на константите, променливите или изразите, които желаем. В QBasic това е операторът PRINT, притежаващ следния най–общ запис:

PRINT списък от параметри

Списъкът от параметри може да съдържа последователност от константи, променливи или изрази, разделени със символите „," или „;". Операторът извежда върху екрана на монитора последователно стойностите на параметрите в списъка. Стойностите на два параметъра, разделени с „;", се отпечатват плътно една до друга. Ако два параметъра са разделени със запетая, то стойностите им се извеждат на известно разстояние една от друга.

Всеки език за програмиране позволява данните да бъдат отделени от алгоритъма. Отделянето на данните от алгоритъма има твърде важно практическо значение, защото дава възможност една и съща програма (алгоритъм) да се изпълни с различни стойности на входните данни, т. е. да решава по-общо поставената задача. Операторът INPUT е средството в QBasic, с помощта на което може да се въвеждат входните данни по време на изпълнението на програмата (алгоритъма). Най-опростеният възможен запис на оператора INPUT е следният:

INPUT "текст"; име на променлива

При изпълнение на оператора компютърът извежда на екрана текста, записан в кавичките, след което спира и чака потребителят (човекът) да въведе стойност от клавиатурата. Въведената стойност се присвоява на указаната в края на оператора променлива, след което изпълнението на програмата продължава.

Обогатихме речника си с две нови думи (команди) от QBasic – PRINT и INPUT, с които ще напишем нова, по-съвършена и полезна програма за намиране лицето на трапец.

Задача 2

Съставете програма, с която може да се пресметне лицето на всеки трапец, на който знаем дължините на двете основи и височината.

```
CLS
   INPUT "Въведи дължината (в см) на едната основа на трапеца:  "; a
   INPUT "Въведи дължината (в см) на другата основа на трапеца:  "; b
   INPUT "Въведи дължината (в см) на височината в трапеца:  "; h
   s = (a  + b) / 2 * h
   PRINT „Лицето S = "; s ; "кв. см"
END
```

Програма 2

Ще отбележим, че данните за трапеца са отделени от програмата и представляват входни данни за нея. Чрез такава програма може да се изчисли лицето не само на трапеца от Задача 1, но и на всеки друг трапец, за който са известни дължините на основите и височината.

Задача 3

Съставете и изпълнете програма, която:

а) въвежда стойности на променливите a и b;

б) разменя стойностите на променливите a и b;

в) извежда (отпечатва) новите стойности на a и b.

Както вече знаем, алгоритъмът за решаването на тази задача изисква въвеждането и използване на трета променлива.

```
CLS
   INPUT "a = "; a
   INPUT "b = "; b
   p = a : a = b : b = p
   PRINT "a = "; a
   PRINT "b = "; b
END
```

Програма 3

Проследете изпълнението на програмата за начални стойности на променливите a = 2 и b = 3.

ВЪПРОСИ И ЗАДАЧИ

1. Съставете програма, с помощта на която изчислете лицата на триъгълниците с дължини на страна и височината към нея съответно:

а) 10,15 м и 7.79 м;

б) 89.87 см и 1.2 м;

в) 1.5 м и 99.6 см.

2. Лицето на триъгълника може да се изчисли, ако са известни дължините на трите му страни. За целта се използва формулата $S = \sqrt{p(p-a)(p-b)(p-c)}$, където с a, b и с са означени дължините на страните на триъгълника, а с p – полупериметърът му. Тази формула се нарича Херонова формула. Съставете програма, с помощта на която изчислете лицата на триъгълници с дължини на страните съответно:

а) 10,15 м, 7.79 м, 8.13 м;

б) 89.87 см, 1.2 м, 99 см;

в) 1.5 м и 99.6 см, 120 см.

3. Съставете програма, която изчислява и извежда стойността на израза:

а) $z = -\frac{(12{,}5 - y)}{2y^4 + 0{,}5} x^2$

б) $z = \frac{(y - 4{,}5)}{\sqrt[4]{y^2 + 1}} x - \sqrt[5]{(x - y)^3}$

в) $z = \frac{\sqrt[3]{(x - y)}}{3\sqrt[2]{15{,}3}} + y\sqrt[4]{4{,}171^3}$

при въвеждане конкретни стойности на променливите x и y от клавиатурата.

4. Съставете и изпълнете програма, която пресмята и извежда големината на правия ток по закона на Ом при входни данни – стойността на напрежението и съпротивлението във веригата.

9.5. ОПЕРАТОРИ ЗА ПРЕХОД

В предходния урок съставихме прости програми, реализиращи линейни алгоритми. Повечето алгоритми обаче са разклонени, имат по-сложна логика, която се описва с помощта на логически условия и елементарни действия за преход. Ето защо езиците за програмиране задължително притежават средства (оператори), с помощта на които се изразяват логически условия и преход. QBasic притежава класически такива.

ОПЕРАТОР ЗА БЕЗУСЛОВЕН ПРЕХОД

GOTO етикет

Операторът GOTO насочва хода на програмата към изпълнение на реда, означен с посочения в оператора етикет.

Пример 1

При изпълнението на програмата:

```
    test = 5
    GOTO et1
    test = 10
et1: PRINT „стойността е“ ; test
```

ще се изведе числото 5, а не 10.

ОПЕРАТОР ЗА УСЛОВЕН ПРЕХОД IF

С оператора IF се описват логическите условия в алгоритмите. IF се нарича оператор за условен преход и притежава две форми – съкратена и пълна.

Конструкцията:

IF условие THEN оператор1 : оператор2 : ... : операторN

представя съкратената форма на оператора IF и има следната семантика:

Проверява се верността на условието записано след IF. Ако условието се окаже вярно, тогава се изпълняват (последователно, от ляво на дясно) указаните след THEN оператори. Ако условието не е вярно, се преминава към изпълнение на операторите, записани в следващия ред в програмата.

Пример 2

а)
```
    prm = 5
    IF prm < 10 THEN GOTO et2
    prm = 15
et2: PRINT “стойността на prm е“ ; prm
```

б)
```
    prm = 5
    IF prm = 10 THEN GOTO et2
    prm = 15
et2: PRINT “стойността на prm е “ ; prm
```

Програмата от подусловие а) извежда чилото 5, а тази от б) – числото 15. Обяснете защо.

Забележка:

Съществува съдържателна разлика в смисъла, който се влага при използването на символа „=“ в математиката и информатиката. В математиката символът „=“ се схваща като „равенство“, докато символа „=“ в информатиката се използва за означаване на оператор за присвояване или за изразяване на отношение в логическите условия.

Един възможен запис на пълната форма на оператор IF е:

IF условие THEN оператори1 ELSE оператори2

където с „оператори1“ и „оператори2“ са означени две последователности, всяка от които съдържа един или няколко оператора, разделени със символа „:“.

Пълната форма на конструкцията IF има следната семантика:

Най-напред се проверява верността на условието, записано след IF. Ако условието се окаже изпълнено (т. е. вярно), се изпълняват

последователно записаните след THEN оператори, след което ходът на програмата (т. е. на алгоритъма) продължава от следващия ред в програмата.

Ако пък условието не е удовлетворено, се изпълняват операторите, записани след ELSE, след което ходът на програмата отново преминава към следващия ред в програмата.

Задача 1

Да се състави програма, която изчислява и извежда стойностите на функцията f(x):

$$f(x) = \begin{cases} \sqrt[4]{x^2 - 5x + 4}, & \text{за } x \le 1 \\ \sqrt[3]{x - 1}, & \text{за } x > 1 \end{cases}$$

От условието на задачата е видно, че стойността на функцията трябва да се изчислява по два алтернативни начина. За стойности на аргумента x, които са по-малки или равни на 1, пресмятането се извършва по формулата $\sqrt[4]{x^2 - 5x + 4}$, а за по-големите – по формулата $\sqrt[3]{x - 9}$.

```
CLS
   INPUT „x="; x
   IF x < = 1 THEN f = ( x*x – 5 * x + 4) ^ 0.25 ELSE f = (x - 1) ^ ( 1 / 3)
   PRINT "f(" ; x ; ")="; f
END
```

Програма 1

У Какъв резултат ще изведе програма 1, ако при изпълнението ú въведем стойности на аргумента:

а) x = 0; б) x = 1; в) x = 17.

Задача 2

Да се състави програма, която изчислява и извежда корените на квадратните уравнения от вида $ax^2 + bx + c = 0$.

Едно възможно решение на поставената задача е направено в представената по–долу програма 2. Обръщаме внимание на факта, че вторият ред на програмата осигурява коректност на входните данни, като не позволява да се достигне до изчисляване на корените на уравнението, когато коефициентът a е равен на 0.

```
CLS
   vha: INPUT "a="; a
        IF a = 0 THEN GOTO vha
        INPUT "b="; b
        INPUT "c="; c
        d = b*b – 4*a*c
        IF d < 0 THEN PRINT "Уравнението няма решение" : END
        IF d = 0 THEN PRINT "x1="; -b/(2*a) : END
        x1 = (-b + d ^ 0.5) / (2 * a)
        x2 = (-b – d ^ 0.5) / (2 * a)
        PRINT "x1=" ; x1 , "x2=" ; x2
END
```

Програма 2

ВЪПРОСИ И ЗАДАЧИ

1. Съставете програма, която отпечатва абсолютната стойност на числата.

2. Съставете програма, изчисляваща стойностите на функцията:

а) $f(x) = \begin{cases} \sqrt[3]{1+x}, & \text{за } x \le 0 \\ \dfrac{1}{2x}, & \text{за } x > 0 \end{cases}$

б) $f(x) = \begin{cases} \dfrac{1}{x+1}, & \text{за } x < -1 \\ \sqrt[3]{x^2 - 9x + 8}, & \text{за } x \ge -1 \end{cases}$

в) $f(x)=\begin{cases}\sqrt[4]{3-x} & \text{, за } x<3\\ \sqrt[3]{1-x^2} & \text{, за } x=3\\ \sqrt[2]{x^2-4x+3} & \text{, за } x\ \ 3\end{cases}$

г) $f(x)=\begin{cases}\dfrac{1}{5\sqrt[3]{1-x^2}} & \text{, за } x<-1\\ \sqrt[3]{1-x^2} & \text{, за } -1\le x<2\\ 1+\sqrt[3]{x^2-3x+2} & \text{, за } x\ge 2\end{cases}$

3. Съставете програма, която открива и съобщава дали дадено цяло число е четно или не.

Упътване: Използвайте функцията INT(x), стойността на която е най-голямото цяло число, ненадминаващо аргумента x.

4. Съставете програма, която разпознава и съобщава коя от годините на 20 век е била високосна и коя не.

Упътване: Използвайте, че високосните години са кратни на 4, както и даденото в предната задача упътване.

5. Съставете програма, която:

а) въвежда различни стойности на променливите a и b;

б) отпечатва името и стойността на тази от променливите a и b, която притежава по-голяма стойност.

6. Съставете програма, която решава:

а) линейни уравнения от вида $ax - b = 0$;

б)* линейни неравенства от вида $ax > b$;

в)* квадратни неравенства от вида $ax^2 + bx + c < 0$;

г)* биквадратни уравнения от вида $ax^4 + bx^2 + c < 0$.

4.5. ЦИКЛИЧНИ АЛГОРИТМИЧНИ КОНСТРУКЦИИ

Интересно и важно практическо значение има възможността за многократно изпълнение на определена група от елементарни действия. Всяко изпълнение на такава група обикновено се извършва върху едни и същи като структура, но различни по стойност входни данни. За да вникнем в същността на цикличните процеси, ще разгледаме по-подробно следния пример:

Задача 1

Опишете с блок-схема и компютърна програма алгоритъм за пресмятане на сумата S = 1 + 2 + 3 + ... + 100.

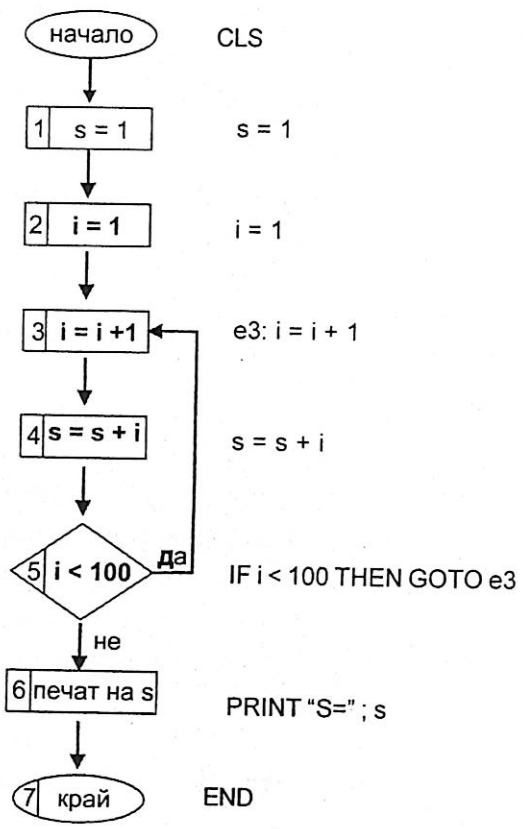
Един подход за решаването на задачата е следният:

Към числото 1 прибавяме 2, към получената сума прибавяме 3 и т. н. продължаваме докато накрая прибавим и числото 100.

Формално алгоритъмът за решаване на задача 1 може да се опише, като се използват две променливи – i и s.

Желаната сума постепенно ще се натрупва и съхранява в променливата s. Следователно, първоначалната стойност на s е 1.

Променливата i трябва да получи стойностите: 2, 3, 4, ... , 100. Всяка една от тези стойности последователно ще се прибавя към текущата стойност на променливата s, което ще доведе до формиране на желаната сума. Чрез фиг. 1 и програма 1 е представен едно възможно решение на задачата.



фиг. 1 Програма 1

Обръщаме внимание на факта, че елементарното действие, записано в блок 3 на фиг. 1, е безсмислено от математическа гледна точка, защото от i = i + 1 следва 0 = 1!! Както вече знаем, в информатиката често се използва подобен запис, но в него се влага всъщност следният смисъл: на променливата i се присвоява старата стойност на i, увеличена с 1.

Да проследим по блок–схемата (фиг. 1) алгоритъма за намиране на сумата S = 1 + 2 + 3 + ... + 100, като записваме номерата на блоковете, които изпълняваме. Ще получим редицата от числа: 1, 2, 3, 4, 5, 3, 4, 5, 3, 4, 5, ... , 3, 4, 5, 6,7. Забелязваме, че групата от елементарни действия с номера 3, 4, 5 се повтаря многократно в хода на алгоритъма. При решаването на тази, а и на много други задачи със средствата на информатиката се налага да се описват повтарящи се дейности или процеси.

Определение: Алгоритмична конструкция, чрез която се организира неколкократно (многократно) изпълнение на група от последователни елементарни действия, се нарича **цикъл**.

Обикновено цикличните конструкции се състоят от три части:

1. **Спомагателна.** В нея се задават началните стойности на променливите, с помощта на които се организира цикълът.

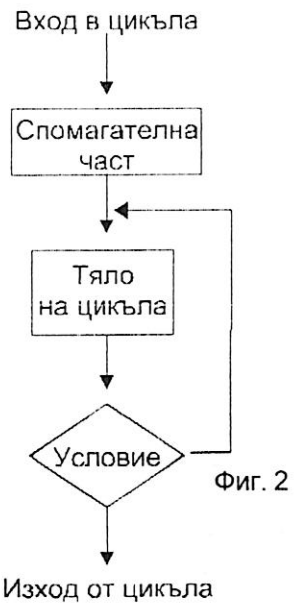
2. **Тяло на цикъла.** Съдържа блока от елементарни действия (оператори), подлежащи на многократно повторение.

3. **Условие за прекратяване на цикъла.** Чрез него се проверява необходимостта от следващо изпълнение на тялото на цикъла.

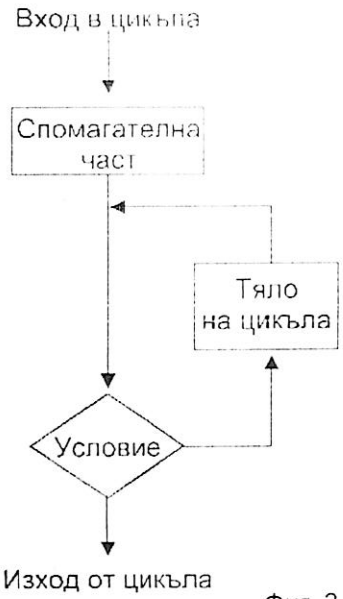
Алгоритъмът, описан с блок–схемата от фиг. 1, съдържа цикъл. Спомагателната част на цикъла са елементарните действия с номера 1 и 2. Тялото на цикъла се състои от инструкции с номера 3 и 4. Логическият блок 5 съдържа условието за прекратяване на цикъла.

Всъщност цикличните конструкции могат да бъдат организирани по един от следните два начина:

Когато цикличната конструкция е създадена (подредена) по начина, показан на фиг. 2, казваме че е организиран **цикъл с постусловие**. Когато подготвителната част, тялото и условието на един цикъл са подредени като на фиг. 3, казваме, че е организиран цикъл с **предусловие**.



Фиг. 2



Фиг. 3

Тук трябва да отбележим, че подредбата на тялото и условието при цикличните алгоритмични конструкции е важна, защото води до следната функционална разлика: тялото на цикъла с постусловие се изпълнява винаги (поне веднъж), докато тялото на цикъла с предусловие може и да не се изпълни при преминаване на алгоритъма през цикъла.

$У_1$ Каква е ролята на действие 4 в алгоритъма, описан с блок–схемата от фиг.1?

$У_2$ Какво би се получило, ако пропуснем блок 1 от блок–схемата от фиг. 1?

$У_3$ Какво би се случило, ако пропуснем блок 5 в блок–схемата от фиг. 1?

$У_4$ Проследете указанията на алгоритъма, описан на фиг.1, за първите няколко стойности на променливата i. Каква ще бъде стойността на s в момента, когато i стане равно на 4?

$У_5$ Определете типа на цикличната конструкция, която е използвана в решението на задача 1.

$У_6$ Колко пъти трябва да се изпълни тялото на цикъла в програма 1, за да се изчисли сумата S = 1 + 2 + 3 + ... + 100?

$У_7$ Редактирайте алгоритъма, описан с блок–схемата от фиг.1, така че да пресмята сумата:

а) S = 2 + 4 + 6 + ... + 100;

б) S = 1 + 3 + 5 + ... + 99.

Задача 2

Нека x е положително число и $a_0 = 1$. Тогава може да пресмятаме едно след друго числата $a_1, a_2, a_3, ..., a_k, ...$ в следната последователност:

$a_1 = 0.5 (a_0 + x / a_0)$

$a_2 = 0.5 (a_1 + x / a_1)$

$a_3 = 0.5 (a_2 + x / a_2)$

...

$a_{k-1} = 0.5 (a_{k-2} + x / a_{k-2})$

$a_k = 0.5 (a_{k-1} + x / a_{k-1})$

...

1. Създайте компютърна програма, която:

а) въвежда положителна стойност на променливата x;

б) пресмята и извежда първото число в редицата a_k, за което е изпълнено неравенството $|a_k – a_{k-1}| \le 0.001$;

2. Изпълнете създадената от вас програма за следните стойности на x: 1, 2, 4, 9, 100. Открийте каква зависимост съществува между стойността на x и резултата, изведен от програмата.

3. Прогнозирайте какъв ще бъде резултатът, ако се изпълни програмата за x = 81.

Дадената по-долу програма 2 представя едно възможно решение на задача 2. Използвана е вградената в QBasic функция ABS (аритметичен израз), която пресмята абсолютната стойност на записания в скобите израз.

```
CLS
    ak = 1
    INPUT "X = "; x
e1: ak1 = ak
    ak = 0.5 * (ak + x  / ak)
    IF ABS (ak – ak1) > 0.001 THEN GOTO e1
    PRINT "ak = "; ak
END
```

Програма 2

В заключение ще отбележим, че съществуват два типа циклични задачи. При едните е възможно повторенията на тялото да се предвидят, организират и зададат още при създаването на алгоритъма. Такава е задача 1.

В задача 2, а и в много други задачи, е твърде трудно (а понякога и невъзможно) да се съобрази предварително колко пъти трябва да се изпълни тялото на цикъла, за да се достигне до решението. В такива случаи повторенията на тялото се контролират и прекъсват в хода на изпълнение алгоритъма с помощта на условие.

ВЪПРОСИ И ЗАДАЧИ

1. Решете задача 1 от урока, като използвате цикъл с предусловие.

2. Решете задача 2 от урока, като използвате цикъл с предусловие.

3. Съставете и опишете с:

а) блок–схема;

б) компютърна програма, алгоритъм, който пресмята (ако е възможно) и отпечатва стойностите на функцията

$$y = \frac{1}{(x-3)\,(x+9)},$$

изчислени в следните възли на аргумента x:

-10; -9,5; -9; -8,5; ... ; -1; -0,5; 0; 0,5; 1; ... ; 9; 9,5; 10.

4. Съставете и опишете с блок–схема и компютърна програма алгоритъм за пресмятане на сумата:

а) $S = 1+\frac{1}{2}+\frac{2}{3}+\ldots\frac{100}{101}$

б) $S = -1+\frac{1}{3}-\frac{1}{5}+\frac{1}{7}-\cdots-\frac{1}{35}$

в)* $S = -1+\frac{2}{3}-\frac{4}{5}+\frac{8}{7}-\frac{16}{9}\cdots-\frac{256}{17}$

5. Съставете компютърна програма, която пресмята и отпечатва сумата:

$$S = 1+\frac{1}{2}+\frac{2}{3^2}+\frac{3}{4^2}+\frac{4}{5^2}+\ldots\left\{\frac{k}{(k+1)^2}\right\}+\cdots$$

само на тези събираеми, за които $\left|\frac{k}{(k+1)^2}\right| \geq 0.001$.

6. Съставете компютърна програма, която:

а) въвежда реална стойност в променливата x;

б) пресмята и отпечатва сумата:

$$S = 1+\frac{x}{2}+\frac{x^2}{3}+\frac{x^3}{4}+\ldots+\frac{x^9}{10}$$

Ч.7. ОПЕРАТОР ЗА ЦИКЪЛ

Цикличните алгоритмични конструкции се използват твърде често за обработка на сходни като структура, но различни по стойност входни данни и междинни резултати. Това е наложило да бъдат разработени и заложени в езиците за програмиране специализирани оператори, с помощта на които бързо, удобно и нагледно се описват циклични конструкции. В по-нататъшното изложение ще разгледаме подетайлно цикличен процес, организиран с помощта променлива, както и реализацията му с едно популярно средство – операторът FOR.

Задача 1

Създайте компютърна програма, която пресмята и извежда сумата:

$$S = 1+\frac{1}{3}+\frac{1}{5}+\frac{1}{7}+\ldots\frac{1}{101}$$

В представеното по-долу решение (програма 1) е използван цикъл с **постусловие** и две променливи – i и s.

```
CLS
   s = 1
   i = 3
sum: s = s + 1 / i
   i = i + 2
   IF i < = 101 THEN GOTO sum
   PRINT "S = " ; s
END
```

Програма 1

Търсената сума S се натрупва постепенно в променливата s. Ето защо за началната стойност на променливата s е прието числото 1.

Ще отбележим, че ходът на цикъла е организиран и се управлява с помощта на променливата i. Елементарните действия в спомагателната част и тялото на цикъла са така дефинирани и подредени, че в процеса на изпълнението им i ще приема последователно стойностите: 3, 5, 7,..., 101.

За всяка една от стойностите на i ще следва изпълнение на тялото на цикъла (операторът s = s + 1 / i), което ще доведе до постепенното натрупване на желаната сума в променливата s.

На практика сумата S ще се формира в хода на изпълнение на програмата в резултат на цикличен процес.

Цикличните конструкции, организирани с помощта на променлива, се използват твърде често като елемент на алгоритмите. Ето защо, повечето езици за програмиране притежават оператор, който организира и реализира циклични процеси с помощта на променлива. В QBasic това е операторът FOR, общият вид на който е:

FOR име на променлива = нач.ст. **TO** последна ст. **STEP** стъпка
  тяло на цикъла
NEXT име на променлива

Цикличната конструкция започва с FOR и завършва с NEXT. Тялото на цикъла се състои от операторите, записани между FOR и NEXT. Допуска се да липсва STEP „стъпка". В такъв случай стъпката се приема за 1.

След FOR и NEXT се записва една и съща променлива, наричана „брояч" на цикъла.

Операторът FOR ... NEXT в QBasic реализира автоматизирано цикъл с постусловие, който се извършва по алгоритъма, представен на фиг. 1.

фиг. 1

Програма 2 представлява едно друго решение на задача 1, което е направено с помощта на нововъведения оператор FOR.

```
CLS
   s = 1
   FOR i = 3 TO 101 STEP  2
      s = s + 1 / i
   NEXT i
   PRINT "S = " ; s
END
```

Програма 2

При изпълнение на цикъла FOR променливата i ще получава най-напред началната си стойност 3, а след това последователно стойностите 5, 7, 9, ... , 101. За всяка една от тези стойности ще се изпълни тялото на цикъла (оператора s = s + 1 / i).

У, Предвидете резултата от следните програми:

а)
```
CLS
  FOR j = 1 TO 99
  NEXT j
  PRINT j
END
```

б)
```
CLS
  s = 0
 FOR j = 5 TO 1 STEP -2.5
    s = s + 1
  NEXT j
  PRINT s
  PRINT j
END
```

в)
```
CLS
  FOR n = 9 TO 1
    PRINT n
  NEXT n
  PRINT n
END
```

ВЪПРОСИ И ЗАДАЧИ:

1. Решете задача от урока, като използвате цикъл с предусловие.

2. Съставете компютърна програма, която пресмята (ако е възможно) и отпечатва стойностите на функциите:

а) $y = \sqrt[4]{x} - \sqrt{(3 - x)}$

б) $y = \sqrt[4]{x - \frac{1-x}{x+3}}$

изчислени във възлите x = -5 + 0.1*i, изчислени за i = 0; 1; 2; ...; 100.

3. Съставете компютърна програма за пресмятане на сумата:

а) $S = \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \frac{1}{8} + \cdots + \frac{1}{50}$

б) $S = 2 + \frac{3}{2} + \frac{4}{3} + \cdots + \frac{101}{100}$

в) $S = 1 - \frac{1}{3} + \frac{1}{6} - \frac{1}{9} + \cdots + \frac{1}{60}$

г) $S = -1 + \frac{2}{3} - \frac{4}{5} + \frac{6}{7} - \cdots + \frac{50}{51}$

Ч 8. ДАННИ

Компютрите са създадени и усъвършенствани от хората с цел да извършват бързо и точно сложни и дълги пресмятания. В повечето приложения на съвременните компютри обаче се използва главно способността им да запомнят, а след това и да дават достъп до огромни количества информация. Това е тяхна основна функция, а извършването на аритметичните операции е важна, но в известен смисъл второстепенна част от компютърната обработка на информацията.

За да бъде приета, запомнена и обработена информацията трябва да се представи и въведе в компютъра под формата на данни. Превръщането на информацията в данни е процедура, която включва няколко успоредни и взаимно свързани дейности.

ОПРЕДЕЛЯНЕ СЪСТАВА И ИЗВЛИЧАНЕ СТОЙНОСТИТЕ НА ДАННИТЕ

Всяка конкретна информационна задача е свързана и се отнася за някаква крайна съвкупност от обекти или процеси. Ето защо най-напред се определят обектите или процесите, които имат отношение към задачата. След това се избира онзи набор от техни характеристики, които имат съществено значение и ще се използват при решаването на задачата. За всеки един обект се извършва измерване, броене или кодиране (по някакво правило) на наблюдаваните характеристики, в резултат на което се получава една многобройна съвкупност от числови или кодирани стойности. Тази съвкупност съдържа **стойностите на данните**, необходими за решаване на задачата.

Ще илюстрираме казаното до тук с дневника на паралелката. В дневника за всеки ученик са записани стойностите на редица важни (от гледна точка на учебния процес) характеристики. Част от тези стойности са получени в резултат на измерване (оценките), броене

(отсъствията), а други в резултат на кодиране (номерата, имената, ЕГН, адресите). Ще отбележим, че тези характеристики на учениците, които нямат отношение към учебния процес (ръст, тегло, цвят на косата и очите, здравословно състояние и т. н.) се смятат за несъществени и не се записват в дневника.

ОПРЕДЕЛЯНЕ ТИПА НА ДАННИТЕ

Спецификата на наблюдаваните характеристики, както и желанието за рационалното разполагане и съхраняване на техните стойности в паметта на компютъра (оперативна и външна) налагат да бъде определяно **множеството от допустими стойности** на данните. Например:

– оценките по отделните предмети са елементи на числовото множество {2, 3, 4, 5, 6};

– средният успех е реално число от интервала [2, 6];

– рожденните дни на учениците са дати от календара.

Множеството от допустими стойности и спецификата на данните определят **операциите**, които се дефинират и е възможно да се извършват с тях.

Ако допустимите стойности на данните представляват множество от числа, то тези данни могат да се обработват с помощта на аритметични операции. Например, ако притежаваме данни, стойностите на които са годишните оценки на учениците от някоя паралелка, то с помощта на операциите събиране и деление може да изчислим средния годишен успех:

– на всеки ученик;

– по отделните предмети;

– на паралелката като цяло.

Когато стойностите на данните представляват последователност от символи, казваме, че тези данни са текстови. Например

имената, записани в дневника на паралелката, са текстови данни за учениците. С текстовите стойности е възможна операцията „конкатенация", която се означава със символа „+". Резултатът от конкатенацията на текстовете T1 и T2 е текст, който се състои от символите на T1, следвани от символите на T2. Например конкатенацията на името „Асен", презимето „Наков" и фамилията „Велев", разделени със символа „интервал" („ ") дава пълното име „Асен Наков Велев", т.е.

„Асен" + „ " + „Наков" + „ " + „Велев" = „Асен Наков Велев"

Ако пък данните са календарни дати, то операцията изваждане (-) е полезна и често се използва. Разликата на две дати е съобразена с особеностите на календара и е равна на броя на дните, между тези дати (т.е. разликата е цяло число). Например:

02.12.2001 – 28.11.2001 = 4, защото месец ноември има 30 дни;
02.11.2001 – 28.10.2001 = 5, защото месец октомври има 31 дни;
02.03.2001 – 27.02.2001 = 3, защото 2001 година не е високосна и м. февруари има 28 дни;
02.03.2000 – 27.02.2000 = 4, защото 2000 година е високосна и м. февруари има 29 дни.

Ще отбележим, че в операциите събиране, умножение, деление и степенуване на дати трудно може да се намери някакъв смисъл и практическа полза. Ето защо подобни операции с дати не се дефинират и използват в практиката.

От казаното дотук стигаме до едно важно понятие, което се използва в информатиката и нейните приложения:

Определение: Множеството от допустими стойности на данните заедно с операциите, които могат да се извършват с тях, определят типа на данните.

СТРУКТУРИРАНЕ НА ДАННИТЕ

Данните ще могат да се обработват автоматизирано, само ако са стриктно подредени по някаква схема (правило). Схемата (правилото), по която се подреждат стойностите на данните, определя **структурата на данните**.

Структурирането на данните представлява важен етап при решаването на всяка информационна задача. Извършва се в съответствие с логиката на отразяваните характеристики и най-вече с целите, поставени в конкретната информационна задача.

Пример 1

Срочните и годишни оценки на учениците са разположени в дневника на паралелката под формата на таблица, подобна на тази от фиг.1:

№ у-к	Име на ученика	Българ. език			Матем.			Англ. език			Физика			Химия			...	Спорт		
		I	II	Г.	I	II	Г.	I	II	Г.	I	II	Г.	I	II	Г.		I	II	Г.
1	Асен Велев	4	5	5	5	5	5	4	4	4	5	6	6	4	6	5	...	6	6	6
2	Антон Колев	6	6	6	5	5	5	5	6	6	4	5	5	4	4	4	...	5	5	5
3	Боряна Янева	5	5	5	4	6	5	6	6	6	4	4	4	4	6	5	...	5	6	6
...		...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...		...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
23	Юлия Николова	6	6	6	4	5	5	6	6	6	4	5	5	5	5	5	...	4	5	5
24	Явор Желев	4	5	5	5	5	5	4	5	5	5	6	6	4	5	5	...	5	5	5

фиг. 1

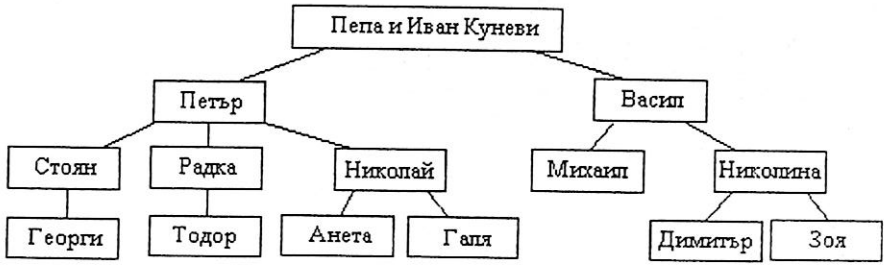
Представянето на данните за срочните и годишни оценки на учениците в структурата „таблица" отразява нагледно резултатите от обучението на всеки ученик и създава предпоставка за автоматизирано изчисляване на средния успех:

– на ученика;
– по отделните предмети;
– на паралелката като цяло,

съответно за първия, втория срок и накрая – за годината.

Пример 2

Ако трябва да опишем роднинските връзки в някоя конкретна фамилия, най-удобно е да подредим данните за родословието в структура, наречена „дърво". На фиг. 2 е представена фамилия, чието начало („корен") произхожда от Иван и Пепа Куневи. Ще отбележим, че в този случай стойности на данните са имена на хора, а чрез схемата (структурата), по която са подредени е зададена връзката „родител" – „дете". Например Петър е баща на Стоян, Николай и Радка и този факт е изразен с помощта на отсечките („ребрата"), които ги свързват в „дървото". Подреждането на данните за родословието в дървовидна структура е удобно средство за представяне на фамилните връзки, защото създава условия лесно, точно и бързо проследяване и определяне на съществуващото родство.



фиг.2

У$_1$ Като използвате данните за фамилия Куневи, представени на фиг. 2, определете роднинската връзка между:

– Иван Кунев и Стоян;
– Пепа и Анета;
– Радка и Васил;
– Анета и Зоя;
– Рада и Михаил;
– Анета и Георги.

В заключение ще направим едно обобщение, касаещо същността на понятието „данни":

Определение: Данните са кодирано или количествено представяне на съществени характеристики на обекти или процеси от реалния свят, направено и подредено за нуждите на конкретна информационна задача.

ЕТАПИ ПРИ РЕШАВАНЕТО НА ИНФОРМАЦИОННИТЕ ЗАДАЧИ

Както вече отбелязахме, всяка информационна задача касае определена, крайна съвкупност от реални обекти и/или процеси. Решаването на информационната задача с помощта на компютър преминава най-общо през няколко взаимно свързани етапа:

1. **Проучване и анализ на задачата**, в резултат на които се определят данните и се създава алгоритъмът за обработката им, такива, че да водят до достигане на поставените цели. Ще отбележим, че алгоритъмът се разработва в съответствие с избраната структура на данните.

2. Замяна на реалните обекти (или процеси) **с данните** за тях.

3. Програмиране на алгоритъма, в резултат на което се създава **приложна програма**, която може да се изпълнява от компютър.

4. **Въвеждане, съхраняване и актуализация на данните** във външната памет на компютъра (дискети, харддиск, CD, магнитни ленти и карти и т. н.). На практика все по често тази дейност се реализира с помощта на някоя готова, специализирана система за управление на бази от данни. Обикновено избраната система се настройва от програмисти за нуждите на конкретната задача, след което се използва от потребителите чрез създадената в етапа 3 приложна програма.

5. **Обработка на данните**. Извършва се автоматизирано от компютър. Данните се въвеждат от външна памет (или клавиатурата) в оперативната памет на „порции" и обработват от процесора, с помощта на операционната система и под управлението на активираната

приложна компютърна програма. Получените междинни и крайни резултати се извеждат и/или съхраняват на подходящи за случая изходни периферни устройства (монитор, принтер, плотер, харддиск, флопи и т.н.).

6. **Интерпретация на получените резултати**, вследствие на което:

– човекът формира отношението, поведението и действията си към отделни компоненти от заобикалящата го действителност;

или

– се управляват автоматизирано, както отделни машини така и цели технологични процеси.

ВЪПРОСИ И ЗАДАЧИ

1. Посочете пример от практиката, в който се използва табличното представяне на данни.

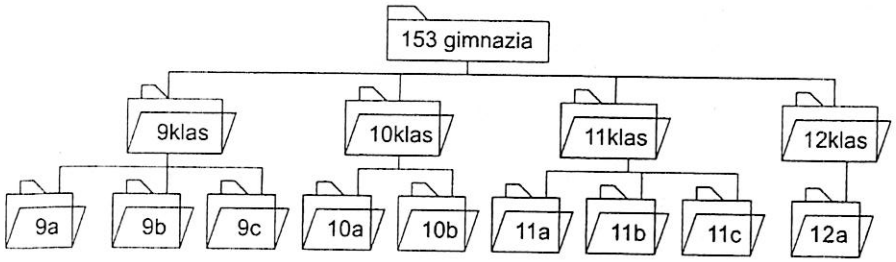
2. Намерете таблица, съхраняваща разстоянията между окръжните градове в страната. Отговорете на следните въпроси:

а) Колко е разстоянието между Шумен и Видин?

б) Кой е най-близкият окръжен град до Плевен?

в) Каква е разликата в разстоянията между София и Бургас ако се пътува през подбалканския път (т. е. през Казанлък) и ако се мине по магистралата през Пловдив?

3. Създайте на дискета следната файлова структура:



ЧАСТ 3. ЕДНОМЕРЕН МАСИВ

Проблемите, свързани с представяне и структуриране на данните, са фундаментални за информатиката. Рационалното решаване на тези проблеми създава предпоставки за реализиране на бърз достъп до отделните компоненти на огромни масиви от данни, а от там и до използване на ефективни алгоритми за обработката им.

Технологичните възможности и ограничения на съвременните компютри и операционни системи, както и разнообразните изисквания, поставени в различни информационните задачи, доведоха до създаване и използване в практиката на широк спектър от схеми за структуриране на данни.

СТРУКТУРАТА „ЕДНОМЕРЕН МАСИВ“

Едно от най-често използваните средства за подреждане, съхраняване и директен достъп до данни е структурата **„едномерен масив“**. Прилага се за описание и обработка на данни, множеството от стойности на които съдържа **фиксиран** брой **еднотипни** елементи.

За да се превърне едно крайно множество от еднотипни стойности в едномерен масив, е необходимо елементите му да се подредят в редица и номерират (последователно). Конструираната по този начин редица от еднотипни стойности се нарича **едномерен масив**. Едномерните масиви се именуват, а номерата на елементите им се наричат **индекси**. Имената на масивите в QBasic се формират както имената на променливите. Всеки елемент на едномерен масив се идентифицира чрез името на масива и поставения му номер (индекс).

Пример 1

Да представим имената на учениците от 9а клас в едномерен масив.

Нека в 9а клас има 25 ученика с имена и номера, както следва: Иван (№ 7), Тодор (№ 24), Валя (№ 3), Калин (№ 18), Ели (№ 5), Ана (№ 1), Юлия (№ 25), Зоя (№6), Борис (№ 2), Галин (№ 4) и т. н.

Нека подредим не самите ученици, а техните имена в съответствие с използваната в паралелката номерация (фиг.1). Получената редица от данни да наречем Names9a:

Names9a:	Ана	Борис	Валя	Галин	Ели	Зоя	Иван	...	Тодор	Юлия
No:	1	2	3	4	5	6	7	...	24	25

фиг. 1

Конструираната редица представлява едномерния масив – Names9a, в които са подредени данни (имената) за учениците от 9а клас. Както вече казахме, всеки елемент от масива се означава и се идентифицира чрез името на масива и поставения му индекс. Например с Names9a(5) се означава и идентифицира петия елемент на масива Names9a. Ще отбележим, че стойността на елемента Names9a(5) е стрингът (трите последователни символа) – „Ели".

По аналогичен начин може да конструираме едномерен масив (фиг. 2) с име Results9a, който съдържа средния успех на учениците от 9а клас:

Results9a:	5.50	4.66	4.33	4.83	5.73	5.13	4.83	...	6.00	4.93
№:	1	2	3	4	5	6	7	...	24	25

фиг. 2

В този пример Results9a (7) е означението на седмия елемент от масива с данни за средния успех на учениците, а стойността на този елемент е числото 4.83.

$У_1$ Посочете името и успеха на ученика с номер 4 в 9а клас.

$У_2$ Какъв е средния успех на Зоя от 9а клас?

$У_3$ Посочете името на ученика с най-слаб успех в класа.

$У_4$ Запишете означенията на тези елементи от масивите, които съдържат данни за момчетата от 9а клас.

ОСНОВНИ ИНФОРМАЦИОННИ ДЕЙНОСТИ С ЕДНОМЕРНИ МАСИВИ ОТ ДАННИ

Основните информационни дейности, които се реализират със структурата масив, са:

1. Въвеждане стойности на елементите.
2. Извеждане стойностите на елементите.

3. Извършване на операции с елементите от масива.
4. Намиране на на-голямата и най-малката стойност в масива.
5. Подреждане (сортиране) не елементите на масива.
6. Търсене в масива.

Реализацията на дейности 1–6 се осъществява с помощта на специфични алгоритми. Част от тези алгоритми ще представим и реализираме, при решаването на няколко последователни задачи.

Повечето езици за програмиране, включително и QBasic, притежават удобни средства за обработка на едномерни масиви от данни. Използването на един масив от данни в компютърната програма започва с описанието (декларацията) му. Описанието включва задължително информация за **типа** и **броя** на елементите в масива, а се осъществява с помощта на специализиран оператор. Декларацията на масивите в QBasic се извършва с оператора DIM (съкращение от DIMENSION– размерност). Например:

DIM Uspeh (25) (1)

е описание на едномерния масив с име Uspeh, който съдържа 26 реални числа. Броят на елементите е 26, защото индексацията на масивите в QBasic се осъществява с числата 0, 1, 2, ...

Декларацията:

DIM Names$ (18) (2)

означава, че в програмата ще бъде използван масив Names, в който могат да се въведат 19 стойности от текстов тип.

Забележка:

Езикът QBasic позволява декларациите на няколко от масивите да се извършат с помощта на един единствен оператор DIM. Например двете декларации, означени с (1) и (2), може да се запишат с оператора:

DIM Uspeh (25) , Names$ (18)

ВЪВЕЖДАНЕ И ИЗВЕЖДАНЕ СТОЙНОСТИ НА ЕЛЕМЕНТИТЕ НА МАСИВ

Задача 1

Съставете програма, която:

а) въвежда имената и средния успех на учениците от вашата паралелка;

б) извежда списък на отличниците в паралелката.

Програма 1 представя едно възможно решение на задачата. Данните за учениците (имена, успех) се въвеждат съответно в масивите Name$ и Result, а броят на учениците – в променливата Br. Решението на задачата преминава през четири последователни етапа:

1. Въвеждане броя на учениците в паралелката.

2. Обявяване на масивите (Name$, Result), които ще се използват в програмата.

3. Въвеждане данните. Имената и успехът се въвеждат в масивите Name$ и Result, последователно (чрез цикъл), в съответствие с номерацията на учениците в паралелката.

4. Извеждане списък на отличниците в паралелката. Списъкът се формира с помощта на цикъл, съдържащ проверка за отличен успех. В списъка ще попадат само учениците, чийто успех е по-голям или равен на 5.50. Ще отбележим, че стойността на променливата j може да се променя в хода на цикъла, като отразява броя на отличните ученици.

```
CLS
  REM Въвеждане броя на учениците
    INPUT "Въведете броя на учениците: "; Br
  REM Описание на масивите
    DIM Name$ (Br), Result (Br)
  REM Въвеждане на данните за учениците
    FOR i = 1 TO Br
      PRINT "Въведи данните на номер: " ; i
      INPUT "Име: "; Name$ (i)
      INPUT "Успех: "; Result (i)
    NEXT i
  REM Извеждане (отпечатване) списък на отличниците
    j = 0
    FOR i = 1 TO Br
     IF  Result (i) >= 5.5 THEN j = j+1 : PRINT j ; ". "; Name$ (i); "-" ; Result
    NEXT i
    IF j = 0 THEN PRINT "Няма отличници"
  END
```

Програма 1

Ще приложим структурата едномерен масив, за да решим с компютър една „интересна" практическа задача, която може да се окаже „полезна" и за вас:

Задача 2

Да се състави програма, която генерира по случаен начин и отпечатва комбинация от числа за ТОТО-2.

За решаването на задачата ще използваме две вградени функции в QBasic:

– „генератор" на случайни числа RND(1). Стойността на тази „функция" е случайно избрано от компютъра число, принадлежащо на интервала [0, 1);

– INT (x). Стойността на тази функция е най-голямото число, което е по-малко или равно на аргумента x.

За да решим задачата, трябва да създадем и опишем алгоритъм, който генерира 6 **различни** цели числа в интервала [1, 49]. Очевидно не може директно да използваме вградената „функция" RND(1), защото тя генерира по случаен начин дробни числови стойности в интервала [0,1).

Нека използваме ограниченията за стойностите на „функцията" RND(1) и направим следните еквивалентни математически преобразувания:

	0	$\leq$ RND(1) < 1	(*49)
=>	0	$\leq$ 49* RND(1) < 49	(INT (x))
=>	INT (0)	$\leq$ INT(49* RND(1)) < INT(49)	
=>	0	$\leq$ INT(49* RND(0)) $\leq$ 48	(+1)
=>	1	$\leq$ 1+ INT (49* RND(1)) $\leq$ 49	

Получаваме аритметичен израз:

1+ INT (49* RND(1)) (3),

стойностите на който са винаги **случайно** избрани **цели** числа, принадлежащи на интервала [1,49].

Алгоритъмът за решаване на задача 2 може да се реализира на компютър с програма 2. „Тегленето" на числата по случаен начин се осъществява в QBasic с помощта на „генератора" за случайни числа (оператор RANDOMIZE) и аритметичния израз: 1+ INT (49* RND(1)). Алгоритъмът използва съществено спомагателния масив a(49), с помощта на които се маркират (с 1) и разпознават вече изтеглените числа, така че да не се допусне повторение.

```
REM Програмата генерира числа за TOTO-2
  CLS
        DIM a (49)
  [REM Нулиране стойностите на a(i)
        FOR i = 1 TO 49
          a (i) = 0
        NEXT i
  REM Активиране на генератора на случайни числа
        RANDOMIZE TIMER
  REM Генериране на 6-те числа
      br = 0
  Gen: RndN = 1 + INT (49* RND(1))
        IF <a (RndN) = 1 THEN GOTO Gen
        PRINT RndN
        a (RndN) = 1
        br = br + 1
        IF br < 6 THEN GOTO Gen
  END
```

Програма 2

Опишете словесно алгоритъма за генериране на числа за TOTO-2.

ВЪПРОСИ И ЗАДАЧИ

1. Измерени са физическите характеристики ръст (в см) и тегло (в кг) на учениците от 9б клас, като са получени съответно следните стойности: Иван (176, 78.5), Петя (166, 51), Никола (180, 75), Оля (170, 67), Катя (161, 55), Росен (190, 85), Антон (172, 67.5), Огнян (179, 73), Юлия (174, 69), Борис (182, 81), Калин (180, 72), Лили (162, 51), Павел (177, 74.5), Зоя (168, 57), Ели (163, 54), Валя (166, 61), Ани (166, 52).

а) Подредете имената, ръста и теглото на учениците съответно в масивите: Imena$, Rst и Kg.

б) Запишете означенията само на тези елементи в масива Rst, които съдържат данните на момичетата.

в) Запишете означенията само на тези елементи на масивите, които съдържат данните за теглото на момчетата.

г) Запишете индексите на тези елементи в масивите, които съдържат данните на най-тежкото момче и на момичето, с най-малко тегло в паралелката.

д) Запишете означенията само на тези елементи в масивите, които съдържат данните на най-ниското момче и на най-високото момиче в паралелката

2. Според лекарите средният ръст на шестнадесетгодишните момчета в България е 174 см, а нормалното тегло е от 62 до 68 кг. Съставете програма, която:

а) въвежда данни за имената, ръста и теглото на група от 10 момчета;

б) извежда списък на тези момчета, ръстът на които е под средната норма за страната;

в) извежда списък на момчета с наднормено тегло.

3. Както вече знаем, изразът 1+ INT (49* RND(1)) може да бъде използван в програмите на QBasic за генериране на цели числа, принадлежащи на интервала [1, 49]. С помощта на вградените функции INT и RND:

а) съставете и запишете друг аритметичен израз, който също може да се използва като „генератор" на числа от TOTO-2.

б) съставете аритметичен израз, стойностите на който са цели числа, принадлежащи на интервала [n, m].

Много често поставените информационни проблеми са твърде сложни, а опитът за прибързаното им решаване обикновено води до загуба на много труд, време и в крайна сметка – до незадоволителни резултати. Ето защо е необходимо да се отдели известно време за внимателно проучване и анализ на проблема, в резултат на което да се формира стратегията за решаването му. Важен етап от разработването на стратегията представлява разбиването на задачата на обособени подзадачи (модули). Раздробяването на „сложното" на части създава условия:

– за по-дълбоко вникване в същността му;

– отделните модули да бъдат разработени от различни подизпълнители, с подходяща квалификация;

– за многократно използване на отделните модули, както в текущата задача, а също така и като градивен елемент при разработката на друга проблематика.

ПОДАЛГОРИТМИ

Модулният подход се използва широко и в областта на информатиката, като намира различни форми на проявление. Така например, всяка информационна задача обикновено се „раздробява" по подходящ начин на определени подзадачи. За решаването на всяка от тези подзадачи се съставя и описва алгоритъм, наричан **подалгоритъм**. Създадените подалгоритми се подреждат в подходяща последователност, която в крайна сметка формира общия алгоритъм за решаването на задачата.

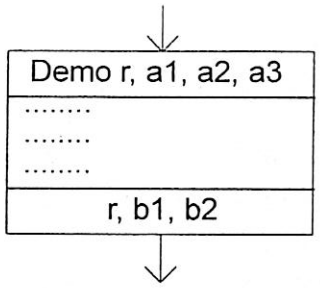
Всеки подалгоритъм има определено място в структурата на цялостния алгоритъм. Освен мястото важни са и връзките (взаимодействието) на подалгоритъма, както с предхождащите го, така и със следващите го части на цялостния алгоритъм.

За да илюстрираме и разясним най-общо механизма на това взаимодействие ще използваме схематичното представяне на един примерен подалгоритъм (фиг. 1), в който ще влагаме следния смисъл:

1. Подалгоритъмът е с име Demo.

2. Четирите променливи: r, a1, a2 и a3 изпълняват функцията на **входни параметри** за Demo. Входните параметри трябва задължително да получат подходящи, конкретни (фактически) стойности преди всяко обръщение към Demo. Подалгоритъмът работи, като използва стойностите на входните си параметри. Ще отбележим, че резултатите от изпълнението на подалгоритъма зависят съществено от фактическите стойности на входните му параметри.

3. Подалгоритъмът Demo решава съответната му подзадача. В хода на изпълнението на подалгоритъма се формират стойностите на **изходните му параметри** – променливите r, b1 и b2. Всъщност, Demo „връща" към главния алгоритъм резултатите от своята работа, като стойности на променливите r, b1 и b2.



фиг. 1

Допустимо е една и съща променлива да бъде използвана, както за входен, така и за изходен параметър. Такива параметри се наричат **входно-изходни**. В подалгоритъма Demo променливата r е входно-изходен параметър.

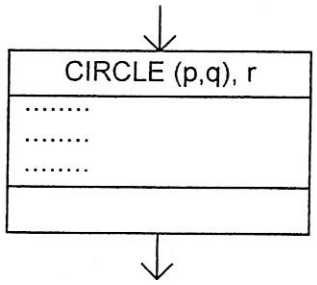
Броят на параметрите за един подалгоритъм зависи от спецификата и потребностите на подзадачата, която решава. В частност, някои подалгоритми могат да нямат съответно входни, изходни или входно-изходни параметри.

Ще отбележим, че чрез параметрите се осъществява взаимодействието между отделните модули на алгоритъма. С други думи казано – входните, изходните и входно-изходните параметри изпълняват функциите на информационен посредник между отделните компоненти на цялостния алгоритъм за решаване на задачата.

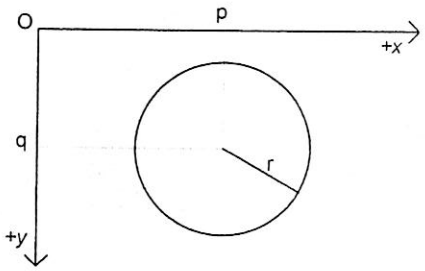
При съставянето и описанието на някои подалгоритми се използват променливи, които имат локално значение и не се достъпни в другите модули, съставящи алгоритъма. Такива променливи се наричат **локални**.

На фиг. 2 е представен схематично подалгоритъм, който чертае окръжност спрямо координатната система от фиг. 3. Входните

параметри (p,q) задават координатите на центъра на окръжността, а r – радиуса.



фиг. 2



фиг. 3

Езикът QBasic притежава оператор, който реализира подалгоритъм за чертане на окръжност. Синтаксисът на този оператор е:

CIRCLE (p , q), r

Параметрите p , q и r са входни параметри и следователно трябва да получат конкретни, фактически стойности при всяка употреба на този оператор.

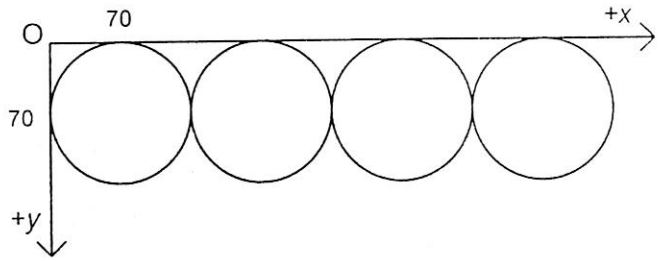
Забележки:

1. Координатната система, спрямо която се чертаят графични изображение в QBasic е подобна на тази от фиг. 3, а центърът й съвпада с горния ляв ъгъл на екрана на монитора.

2. Преди да се използва операторът CIRCLE, трябва задължително да се укаже еднократно типа на монитора, с който работим. Типът на монитора се определя с оператора SCREEN. В понататъшното изложение ще използваме един стандартен тип монитори, които се задават със SCREEN 12. При този тип монитори допустимите стойност по оста x са в интервала [0,640], а тези по оста y – в интервала [0,480].

Задача 1

Като използвате оператора CIRCLE, съставете програма на QBasic, която чертае фигурата:



фиг. 4

Решението на задачата ще сведем до четирикратно обръщение към подалгоритъма CIRCLE. Всяко активиране на подалгоритъма (оператора) CIRCLE ще се предхожда от задаване на подходящи стойности на входните параметри p, q и r. Програма 1 представя линеен алгоритъм за решение на задача 1, докато програма 2 решава задачата с помощта на цикъл.

```
CLS
   REM Задаване типа на монитора
   SCREEN 12
   p = 70 : q = 70 : r = 70
   CIRCLE (p, q), r
   p = 210 : CIRCLE (p, q), r
   p = 350 : CIRCLE (p, q), r
   p = 490 : CIRCLE (p, q), r
END
```

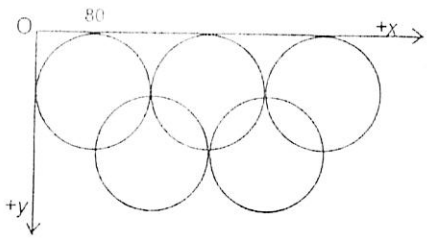
Програма 1

```
CLS
   REM Задаване типа на монитора
   SCREEN 12
   q = 70 : r = 70
   FOR p = 70 TO 490 STEP 140
      CIRCLE (p, q), r
   NEXT p
END
```

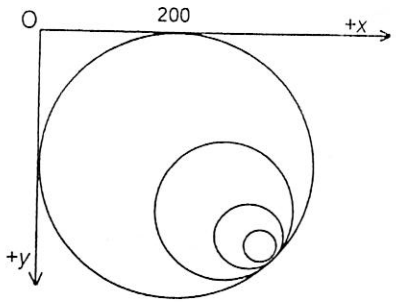
Програма 2

Редактирайте програма 1 така, че да чертае системата от окръжности, представена на:

а) фиг. 5; б)* фиг. 6



фиг. 5

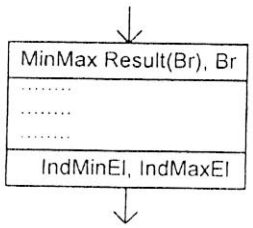


фиг. 6

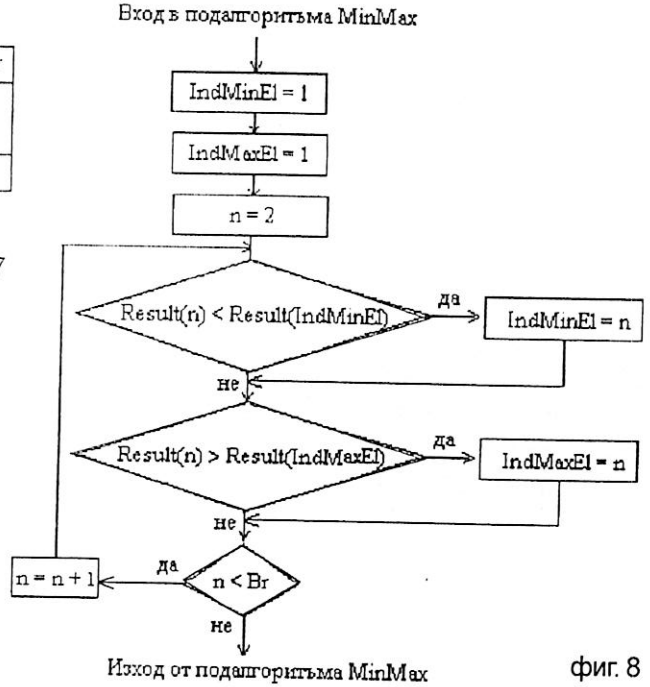
Задача 2

Съставете и опишете с блок-схема подалгоритъм, който намира най-малката и най-голямата стойност на данни, подредени в едномерен масив.

На фиг. 7 и фиг. 8 е представен подалгоритъм, който намира индексите съответно на елемента с най-малката и на елемента с най-голямата стойност в едномерен масив.



фиг. 7



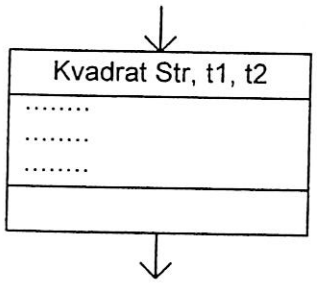
фиг. 8

Входните параметри на MinMax са числовият масив Result и променливата Br, отразяваща броя на елементите му. Променливата IndMinEl се използва за намиране и съхраняване индекса на този елемент от в масива Result, който има най-малка стойност. Аналогично в променливата IndMaxEl се формира индексът на елемента с най-голяма стойност. Стойността на двете променливи IndMinEl и IndMaxEl се получават в резултат на цикличен процес, организиран чрез променливата n. Променливите IndMinEl и IndMaxEl са изходните параметри на подалгоритъма MaxMin, докато променливата n има локално предназначение.

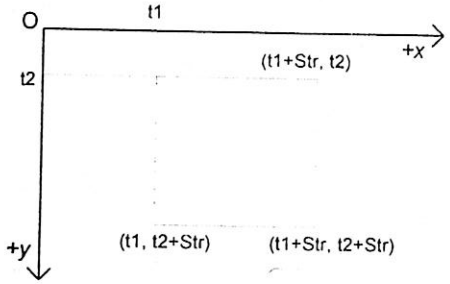
Обръщаме внимание на факта, че подалгоритъмът MinMax намира и връща индексите (т. е. мястото) на минималния и максималния елемент в масива Result, а не най-малката и най-голямата стойност на въведените в него данни. Следователно, след изпълнение на подалгоритъма MinMax, най-малката и най-голямата стойност в данните се намират съответно в елементите: Result(IndMinEl) и Result(IndMaxEl) на масива Result. Ще отбележим, че чрез индексите получаваме директен достъп до екстремалните стойности на масива, което разширява значително възможностите за приложение на подалгоритъма MinMax.

ВЪПРОСИ И ЗАДАЧИ

1. На фиг. 9 е представен схематично подалгоритъмът Kvadrat, с помощта на който може да се чертае квадрат със страни, успоредни на координатните оси Ox и Oy (фиг. 10). Входният параметър Strn задава дължината на страната на квадрата, а t1 и t2 – координатите на горния му ляв връх.

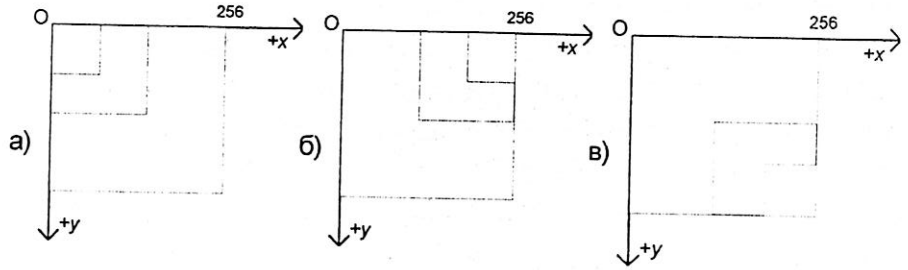


фиг. 9



фиг. 10

Като използвате подалгоритъма Kvadrat, съставете блок-схема на алгоритъм за чертане системата от вложени квадрати:



Забележка: Страната на всеки по-вътрешен квадрат е два пъти по-малка от страната на предходния му.

3. Кое неравенство:

а) IndMinEl ≤ IndMaxEl;

б) IndMinEl > IndMaxEl;

в) IndMinEl ≠ IndMaxEl;

г) Result (IndMinEl) < Result (IndMaxEl);

д) Result (IndMinEl) ≤ Result (IndMaxEl)

е винаги вярно след изпълнение на подалгоритъма MinMax?

Ч.11. ПОДПРОГРАМИ

В езиците за програмиране съществуват правила и средства, с помощта на които се описват и изпълняват подалгоритми. Подалгоритмите, които извършват действия, обикновено се оформят като **подпрограми**. Подалгоритмите, които са предназначени да изчисляват и връщат някаква стойност, най–често се описват и използват като **подпрограми–функции**. Последователността на основните дейности (подалгоритми), които водят до решаване на задачата, се описват в т. нар. **главна програма**. Текстът на главната програма, заедно с текстовете на подпрограмите, съставят цялостния текст на компютърната програма.

Езикът QBasic притежава удобни средства за модулно изграждане на компютърната програма. Отделните модули се отделят и разработват като самостоятелни програмни части, а се активират от главната програма в съответствие с логиката на алгоритъма.

В по-нататъшното изложение ще разгледаме и използваме само едно, несложно за употреба, средство за обръщение и изпълнение на подпрограми в QBasic, а именно операторът с общ вид:

GOSUB етикет

Операторът GOSUB осъществява преход към подпрограма, оформена като самостоятелна програмна част и разположена след главната програма. Всяка такава подпрограма започва с етикет и завършва с оператора RETURN. Етикетът на подпрограмата указва входната точка, т. е. редът, от който започва. Етикета ще използваме както за именоване, така и за обръщение към подпрограмата. Препоръчва се подпрограмата да започва с оператор за коментар (REM), в