КОМПЮТЪР ЗА ВАС

Издание на ЦК на ДКМС

3-4 '89 Година пета Цена 1,60 лв.

ISSN 0205 1893

- ВИРУСНАТА ЕПИДЕМИЯ — У НАС И В ЧУЖБИНА
- ДЕЦАТА В ИНФОРМАЦИОННИЯ ВЕК
- ПРЕДСТАВЯМЕ ВИ
  — IBM PS DOS 4.0
  — COMPILER PLUS
  — CP/M

Сканиране и обработка:

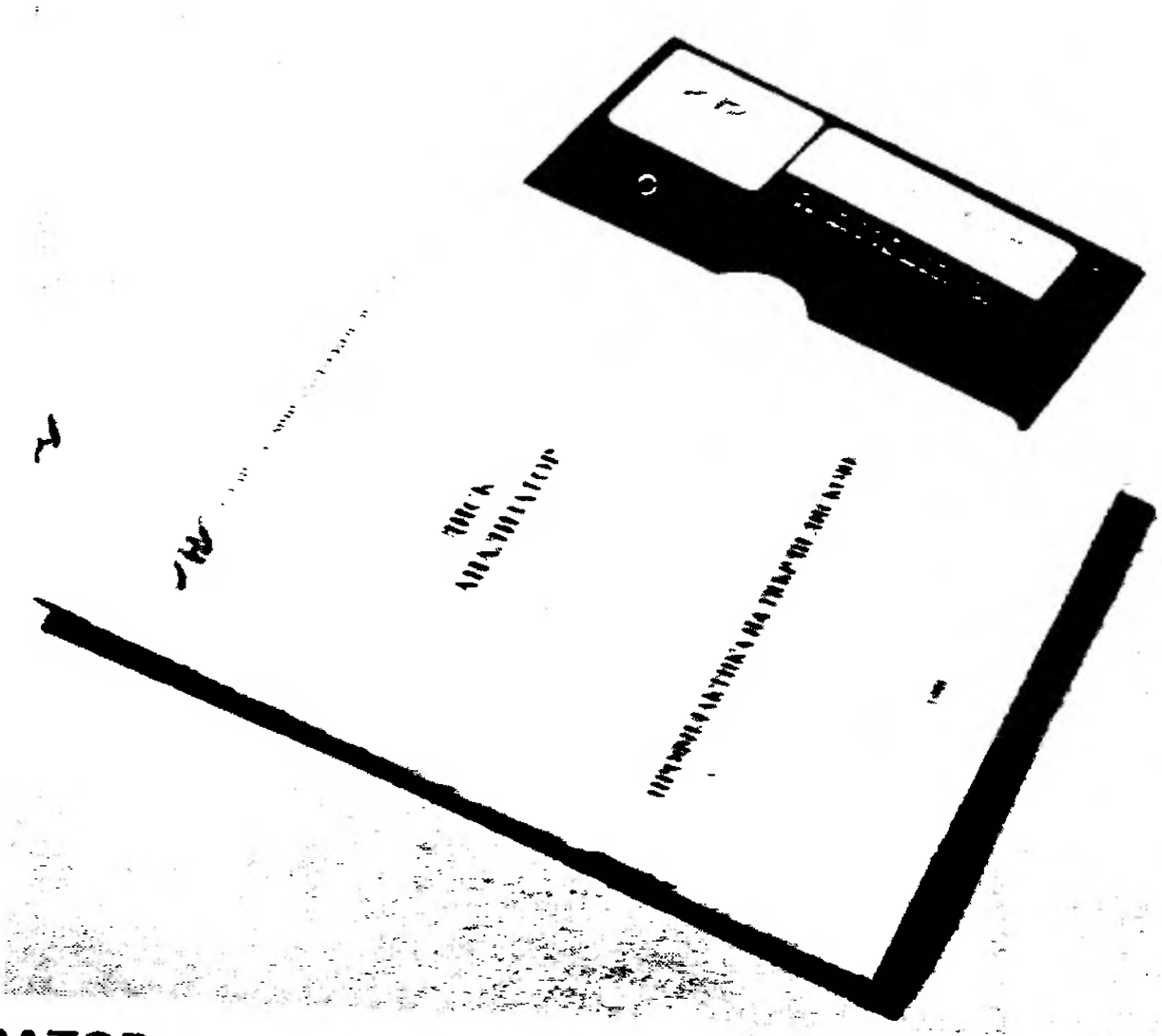
Антон Оруш

www.sandacite.net
deltichko@abv.bg
0896 625 803



Форум
САНДЪЦИТЕ

Всеки 20-мегабайтов твърд диск съдържа над 200 милиона бита информация. При промяна на само един от тях е възможно да се загуби цялата записана информация. А това може да се случи всеки момент и по най-различни причини - при аварийни ситуации, като токов удар, отпадане на напрежението, механично сътресение или в резултат на неизбежни явления при експлоатацията - термична деформация и изместване на главите, остаряване и износване на магнитното покритие, "избледняване" на записа и т.н.



ДИСК-АНАЛИЗАТОР тества всеки бит от работната повърхност на диска и при необходимост използва специални процедури и алгоритми, за да възстанови, докато това е все още възможно повредените записи. Извършва физическо форматиране на повредените пътечки, като съхранява и наново записва намиращата се върху тях информация. При механично разрушаване на някои писти, той записва запазената информацията на друго място на диска, а повредената писта маркира като негодна за ползване.

ДИСК-АНАЛИЗАТОР поддържа файл с информация за текущото състояние на твърдия диск.

ДИСК-АНАЛИЗАТОР позволява да се определи оптималния фактор за последователността при подреждане на секторите на диска и може да извършва физическото му преформатиране за максимално ускоряване на достъпа до информацията.

ДИСК-АНАЛИЗАТОР съдържа специална паркираща програма, която автоматично премества главите върху неизползваема писта, ако не е имало обръщение към диска в интервала между 1 и 15 секунди - по избор.

ДИСК-АНАЛИЗАТОР струва 150 лв.

ПРОИЗВОДИТЕЛ
ТК "КОМПЛЕКТАЦИИ
И ПРОГРАМНО-ТЕХНИЧЕСКИ
УСЛУГИ"
[illegible]
[illegible]
[illegible]

Антивирусни програ-ми, за които разполага-ме с информация към 17 април 1989 г

AIDSTEST	EXE	15118
NTI	EXE	11824
ANTIDOTO	EXE	2240
ANTIVIR	EXE	8560
ANTIVIRP	EXE	9776
ANTIVIRU	EXE	14320
ANTIVRUS	EXE	8560
ANTI4US	EXE	23179
ANTI4US	LOG	17
ANTI4US	DOC	4373
ANTI4US2	EXE	23179
AVIR	EXE	7856
B10	ASM	7138
B10	COM	1196
B11	ASM	6756
B11	COM	956
CURE	EXE	12538
DDV	ASM	2185
DDV	COM	858
DDV1	COM	778
DETECT	EXE	14208
DETECTOR	EXE	13420
DEVIRUS	COM	12302
DEVIRUS	DOC	1443
DIAG	EXE	11834
DIAGCURE	HLP	3048
DOCTOR	EXE	5136
EF	EXE	7264
FDV	EXE	5472
GLOBAL	COM	799
HEBREW	TXT	1508
LOG	ASM	26586
LOG	COM	1807
LOG	DOC	2045
LOOK	EXE	10496
README		39847
README	VD1	3359
SERUM2	EXE	14112
SHOWBALL	ASM	1677
SHOWBALL	COM	232
TEST	COM	25
TP04KILL	EXE	9370
TP33KILL	COM	2746
VACCINE	ASM	3013
VACCINE	COM	571
VACCINE	DOC	13952
VACCINE	REM	1821
VACCINO	COM	373
VIR	EXE	2783
VIRCHK	EXE	9203
VIROLOG	BLG	2773
VIROLOG	DOC	2782
VIROLOG	EXE	10736
VIRUS	BIB	2927
VIRUS D1	EXE	10883
VIR BÊT	EXE	10141
VIRUS	EXE	5984
VIRUS	PAS	1686
VIRUS2	BAS	709
VIRUS2	EXE	32259
VIRUV	EXE	9632
VSHELL	ASM	2112
VSHELL	COM	143
VSHELL	ASM	2112
VSHELL	COM	143

Драги читатели,

В миналия брой обещахме да ви съобщим името на победителя в ПРЕМИЯТА НА ЧИТАТЕЛЯ, организирана от издателство "Техника", списание "Компютър за вас" и НПК по микропроцесорна техника, Правец. Условието за спечелване на премията бе да се отговори на пет въпроса от областта на статистическата обработка на данни, които бяха поместени на последната страница на първия сборник с подбрани материали и програми "Най-доброто от списание "Компютър за вас". В определения срок се получиха общо 109 отговора, от които за верни (виж на стр. 63) бяха приети 5. Между петимата бе теглен жребий и **персоналният компютър Правец-8Д се падна на Росен Маринов Георгиев**, ученик първи курс в техникума по електротехника "М. Ломоносов", Горна Оряховица. Той е петнайсетгодишен и от 1986 година работи на компютър Правец-82.

Първата награда Правец-82 не получи никой, защото авторите на книгата бяха включили известна уловка в петия въпрос, който позволява неоднозначен отговор. Така че напълно изчерпателен отговор никой не можа да даде.

Останалите участници, чиито отговори бяха приети за верни са:

* Станимир Колев, Стара Загора
* Милен Стоянов, Бургас
* Димитър Ковачки, София
* Желязко Терзийски, Пловдив

Техните награди са набор книги по компютърна техника на издателството и безплатен абонамент за списание "Компютър за вас" за 1989 г.

Остана открит въпросът за съдържанието на следващия сборник "Най-доброто от списание "Компютър за вас". Едно е сигурно - и той ще бъде придружен с дискета (или касета), което ще позволи публикуването и на по-дълги програми. За да вземем най-правилното решение, очакваме вашето мнение, драги читатели. Какъв да бъде сборникът?

* Да бъде насочен само към един или към повече типове компютри?
* По какъв признак да бъдат подбрани програмите - игри, помощни програми, с практическо приложение?
* Сборникът да съдържа само софтуер, хардуер или и проблемни статии?
* Да съдържа ли справочни материали за компютри, за операционни системи или програмни езици?
* Уместно ли е да публикуваме кратки ръководства за работа с широкоразпространени (без литература за тях) програмни продукти?

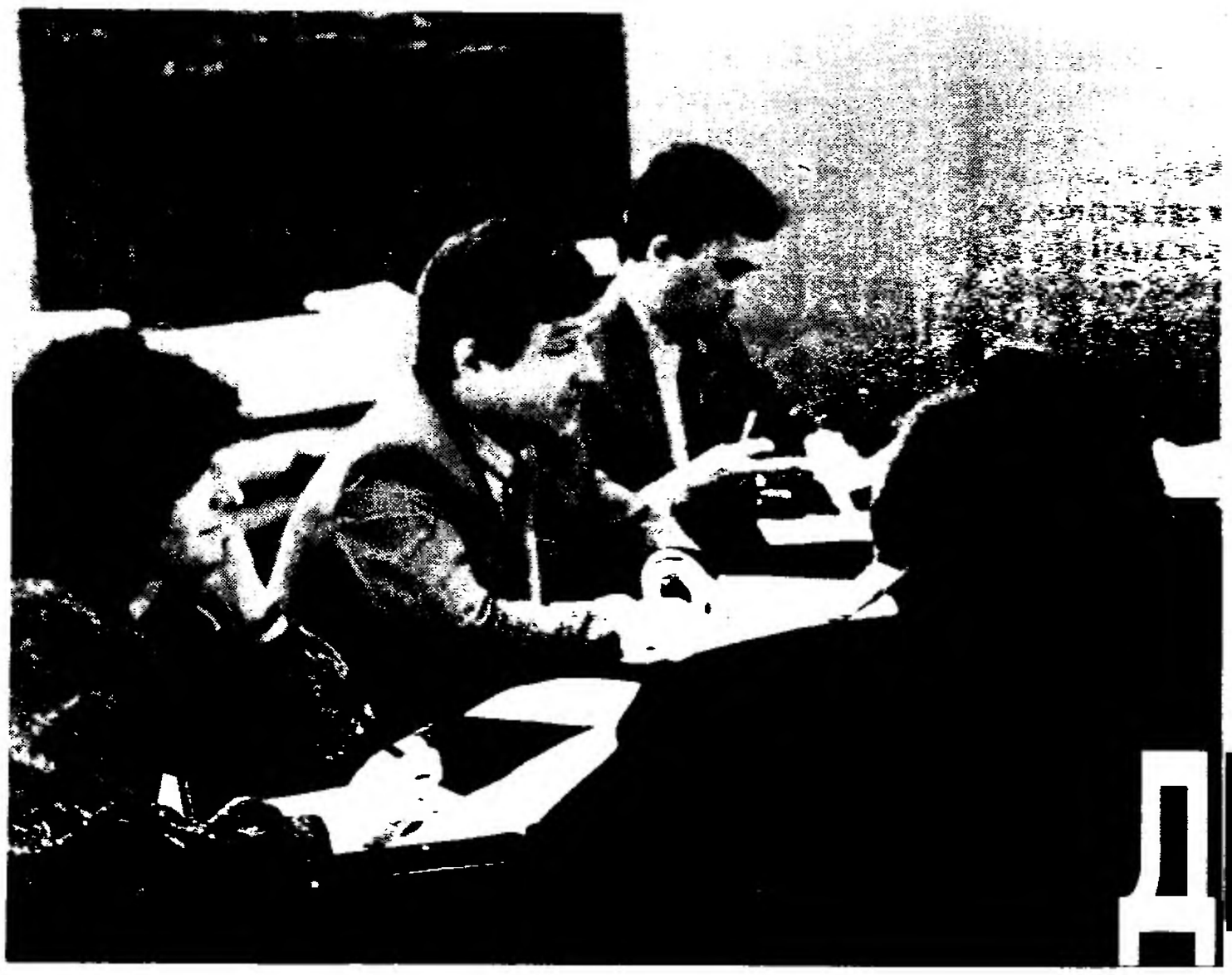
Вече доста се понасъбраха програмите от т.нар. "акция дарителство", които поради голямата им дължина не публикуваме, но записваме безплатно на всеки, който дойде в редакцията. Може би е време те да излязат също в отделен сборник с дискета?

Вариантите за съдържанието на следващите сборници "Най-доброто от "Компютър за вас" не са малко и затова ще очакваме вашите писма с предложения и мнения.

Темата за **компютърните вируси** отново присъства в списанието, тя очевидно и занапред няма да загуби своята актуалност. В предния брой съобщихме, че редакцията разполага с програма за лечение на вируса VHP-648 и с програма-ваксина срещу него, създадени от наши автори. Вече няколкостотин души ни посетиха с дискета в ръка да им запишем антивирусните програми. Понеже темповете на отпечатване на списанието безнадеждно изостават от бързината, с която се променя обстановката на компютърния фронт, към днешна дата, 17 април, вече са необходими две дискети, за да се побере колекцията от антивирусни програми, събрани от нашия консултант по проблемите на вирусите *Веселин Бончев*.

Трябва да се знае обаче, че почистването на твърд диск от вируси съвсем не е елементарна работа и както и самолечението при хората, некомпетентната намеса и използването на програми, които не се познават добре, може да доведе само до лоши резултати. Затова, ако пострадате от вируси и не сте уверени, че можете сами да се справите с тях, обърнете се към редакцията ни. Нашите консултанти са готови да ви помогнат.

Георги Балански



Децата в информационния век

ПРЕД ДИСПЛЕИТЕ НА БЪДЕЩЕТО

Всяка нечетна година непосредствено преди 24 май у нас се провежда международната конференция „Децата в информационния век". Конференцията, която ще се проведе от 20 до 23 май 1989 година в София, е трета от поредицата, чието начало бе поставено през 1985 г. Нашето списание, връстник на този международен форум, активно сътрудничи при подготовката и провеждането му. От 4 до 7 януари тази година в гр. Правец заседава Международният програмен комитет за третата конференция. Във връзка с това събитие екип от редакцията се срещна с Бранимир Ханджиев, организационен секретар на конференцията.

КВ: — Бихте ли разказали на нашите читатели как протича подготовката на третата международна конференция?

— Преди всичко искам да благодаря, че вашето авторитетно списание проявява интерес именно към подготовката. Другите журналисти идват едва на откриването на самата конференция и се интересуват само от проблемите на информатиката в образованието.

Международният програмен комитет се събира два пъти преди провеждането на всяка конференция, но подготовката започна още на втората конференция, когато беше предложена подтемата на тази година — „Развитието на човечеството и възникващите технологии". Първото подготвително заседание на Международния програмен комитет за третата конференция се състоя през май миналата година във Варна. На заседанието в Правец през януари вече беше подготвена окончателната структура на научната програма.

В края на февруари в Холандия ще се срещнат организатори на подобни прояви от цял свят. Поканен е и нашият организационен комитет. Така се надяваме и нашата конференция да заеме достойно място сред световните прояви от този род.

В момента подготвяме второто съобщение за участие, което ще разпространим през февруари.

КВ: — Разкажете за структурата и задачите на Международния програмен комитет.

— Международният програмен комитет с председател академик Благовест Сендов е мозъчният тръст на конференцията. Той се събра в Правец тази година за четвърти път. В него участват видни учени от цял свят — проф. д-р Асен Жабленски — председател на Българската медицинска академия в София, проф. У. Брауер от Техническия университет в Мюнхен, проф. П. Болерслев от Дания, проф. Т. Вашко от Международния институт по приложен системен анализ, проф. А. Семьо-

от Московския държавен университет, проф. А. Мънди-Касъл от Зимбабве, д-р Б. Колис от Канада, д-р К. Бел от Великобритания, проф. С. Чери от Италия, д-р С. Чарп от САЩ — хора, които са не само големи учени, но и големи ентусиасти. Спомням си студения януари на 1985 г., когато София беше побеляла от сняг. За да се включи в работата на Международния програмен комитет, пристигна дори и известният италиански професор Сачердоти, който е инвалид. През двата дни, в които се провеждаха заседанията, го транспортирахме в инвалидната му количка. Той остана верен на конференцията, макар и със задочно участие, като и до днес получаваме неговите писма от Италия.

Изпратихме нашето първо съобщение за третата конференция на 3000 адреса в целия свят. Един от участниците в Международния програмен комитет — д-р Крис Бел от Великобритания, със собствени средства отпечата и разпрати 500 такива съобщения (идентични!) на видни английски учени, без да сме му изпратили материалите във вид „камера реди".

КВ: — Кои организации финансират конференцията?

— Основен организатор и спонсор на конференцията е Министерството на културата, науката и просветата. Съорганизатори и спонсори са ЮНЕСКО, Международната фондация „Людмила Живкова", Международният институт по приложен системен анализ (МИПСА) и Международната федерация по обработка на информацията (ИФИП).

КВ: — Какво е новото тази година?

— Новото се състои в принципно различния подход при подбора на материалите. Докато навремето ние се стараехме по най-различни канали да популяризираме конференцията и да утвърждаваме нейното име в България и по света, включително и от страниците на „Компютър за вас", днес тя се радва на широка популярност и интерес сред много учени от различни водещи центрове по света, които се занимават с тези проблеми. При подготовката на първата конференция се стремяхме да привлечем повече участници с утвърден авторитет. Днес, при подготовката на третата конференция, Международният програмен комитет работи като жури, което много строго пресява изпратените доклади и има възможност да избира измежду много предложения. Дори някои доклади на авторитетни чуждестранни участници бяха отклонени за така наречените постър сесии.

Нашата конференция има собствено направление, но не обсъжда едни и същи проблеми. На всяка нова среща имаме различна подтема, под чийто знак преминава конференцията. През 1985 г. тя беше „Утрешните проблеми днес", две години по-късно — „Възможности за творчество, рационализации и нови дейности", а тази година мотото ще бъде „Развитието на човечеството и възникващите технологии", така че докладите, които приемаме, ще отговарят на това заглавие.

КВ: — Бихте ли се спрели по-подробно на участниците и тематиката, застъпена на третата конференция?

— Конференцията ще заседава в различни секции по следните направления:

- Развитие на човечеството чрез информационните технологии.
- Постиженията на национални и международни проекти за компютризация в образованието.
- Възникващите нови технологии и връзката им с образованието.

Ето и някои от поканените докладчици: Рис Гуин от Великобритания, Нийл Ериксен от Дания, Брайън Рей от Кения, Пер-Ерик Майкълсън от Швеция, Яо Оливейра и Патриша Грийнфилд от САЩ, Джудит Хамънд от Австралия, Елад Пелед от Израел и др.

КВ: — Разкажете за съпътстващите конференцията прояви.

— Още първата конференция показа, че може да прерасне в международна изследователска програма с много други съпътстващи събития. Например по време на втората конференция за първи път се срещнаха редакторите на различни издания с такава насоченост и „Компютър за вас" взе дейно участие. Такава среща ще се състои и през май тази година.

Стана традиция преди всяка конференция у нас да се провежда и съвещание на ИФИП — Международната федерация по обработка на информацията, която също е наш спонсор. По време на втората конференция се състоя и първото състезание по информатика за деца с международно участие. Тази година с помощта на ЮНЕСКО то ще прерасне в международна олимпиада по програмиране за деца, която ще бъде на 17 и 18 май в Правец. Нещо повече — лятната школа за деца в Перислав Залески, която се организира от Института по програмни системи на АН на СССР, реши първите трима, класирали се на олимпиадата, да участват безплатно в тази школа. Предвиждаме и курс за подготовка на кадри по информатика от развиващите се страни.

Има идея изложбата ВИДЕОКОМП, организирана от ВМЕИ „Ленин" (първата изложба беше през 1987 г.), да бъде включена също в международната програма „Децата в информационния век", като тя ще се провежда всяка четна година.

КВ: — С какво допринесе организирането на тази конференция за приложението на информатиката в образованието у нас?

— Благодарение на конференцията можахме да привлечем вниманието на ЮНЕСКО и ПРООН — програмата на ООН за развитие, и да създадем и оборудваме Центъра по информатика в образованието към БАН.

Изследователската програма „Децата в информационния век" е българският принос към международната програма на ЮНЕСКО по информатика. Заместник-председател на бюрото на тази международна програма и председател на нашата изследователска програма е академик Благовест Сендов. В нейните рамки се изпълняват редица проекти, финансирани от ЮНЕСКО. Един от тях е съвместният българо-съветски проект, включващ и курса по информатика, в който ще участват представители от петнайсет развиващи се страни. Резултатите от работата по тази изследователска програма и на Центъра по инфор-

⇒

Е 03—04 89

матика в образованието към БАН ще бъдат докладвани на Международния конгрес по приложението на информатиката в образованието на ЮНЕСКО тази година.

По инициатива на Международната фондация „Людмила Живкова“ и съвместно с ЮНЕСКО в рамките на програмата „Децата в информационния век“ се организира нов международен проект, в който участват учени от 14 страни — Австрия, България, СССР, Канада, Китай, Израел, ФРГ, Франция, Унгария, Япония, Мексико, Швеция, Великобритания и Зимбабве. Той е насочен към изследване на влиянието на компютрите и информационните технологии от медицинска, психологична и социологична гледна точка върху развитието на децата от различни националности и културни. Изследванията по този проект също ще бъдат докладвани на специално заседание на конференцията от проф д-р Асен Жабленски и проф. Бети Колис (Канада) — нов член на Международния комитет.

КВ: — И накрая — какви трудности срещате при подготовката на конференцията?

— Първата конференция беше замислена като еднократна проява от група ентусиасти, които на един осембитов компютър събираха имена и адреси на потенциални участници. Сега, когато тя извоюва международен авторитет, нещата не са се променили много. Ентусиазмът на нашия местен Организационен комитет остава основният двигател за преодоляване на всички технически трудности. Той е съставен от хора, чиито основни задължения са други, а нямаме и техника.

Въпросът за липсата на компютри и периферия за тях е действително въпрос на чест за един организационен комитет на конференция с подобна насоченост. Нашите западни колеги дори не подозират, че ни поставят на колене с простото изискване да им предоставим материали по телефакса. Изобщо има какво да се направи в това отношение.

КОМПЮТЪР ЗА ВАС

□

В началото на годината в гр. Правец заседава Международният програмен комитет на третата конференция „Децата в информационния век“. Обсъдена бе научната програма на подтемата на конференцията — „Развитието на човечеството и възникващите технологии“. В заседанието взеха участие авторитетни чужди и български учени, между които проф. д-р Асен Жабленски, председател на Българската медицинска академия. Той бе тъй любезен да отговори на въпроси на представител на редакцията на КВ.

КАКВО ПРАВЯТ КОМПЮТРИТЕ С ДЕЦАТА?

КВ: — По време на втората международна конференция „Децата в информационния век“ експертите от ЮНЕСКО поставиха един важен въпрос — необходимостта от задълбочени научни изследвания за въздействието на компютризираната учебна среда върху интелектуалното и физическото развитие на децата. Професор Жабленски, по време на третата конференция ще бъде обсъждан проектът за изследване на влиянието на информатиката върху деца с различна възраст, бит, култура и национален. Какво можете да кажете на тези предварителни етапи в подготовката на конференцията?

— Целта на изследването е да се проучат преди всичко психологическите и социалните страни на въздействието на новата педагогика, свързана с информатиката в класната стая, с влиянието на културата и семейството върху психическото развитие на детето до училищна възраст. Ще се сравняват резултатите от различни страни, всяка със своите традиции и култура, за да бъде извлечен опит за по-рационално използване на информатиката като инструмент на педагогическия процес. Изследването ще бъде ориентирано към т. нар. второ поколение системи, използващи компютрите за образователни цели. При първото поколение се наблягаше на усвояването на конкретен учебен материал с новите технически средства. С второто поколение системи трябва да бъде създадена новата педагогическа среда като форма на взаимоотношение на детето с околния свят. Следователно тук най-важното са не компютърът и програмата му, а отношението, което става в съзнанието на детето и на учителя.

КВ: — Вредни ли са вече по-добре изследвания в страни с по-голям опит в по-продължителна употреба на компютрите в образованието?

— В няколко такива страни се проведоха сходни изследвания, но те са започнали неотдавна, така че за вредния от тях ефект не може да се говори.

КВ: — Как на практика ще се осъществи новото изследване?

— Предвижда се да участват деца от две различни възрастови групи — едните на 8 г., а другите — на 13—13,5 г. Очакваме да се включат доста страни, сред които Дания, Австрия, ФРГ, СССР, Япония, Китай и САЩ. Интересът е голям. Сега все още сме в подготвителен етап, а заседанието на Международния комитет цели да уточни методиката на проучване, начините, по които ще бъдат избирани децата, стандартизацията на различните фактори, контролирани при такова изследване. Всичко това е абсолютно необходимо, за да могат резултатите на различните страни да бъдат съпоставими, т. е. достоверни.

Проучването ще продължи няколко години. Децата ще бъдат изследвани многократно през цялата учебна година, за да се уточни какво е влиянието на новите методи на обучение върху тяхното интелектуално и социално развитие, върху емоционалния им живот, върху начина, по който те общуват с другите деца — как протича при тях процесът на социализация, стимулира ли компютърната среда тяхното творческо въображение и мислене или, напротив, ги прави по-стереотипни. Това са важни въпроси, които все още нямат отговор. Ние искаме да видим и узнаем какво става с тези деца.

КВ: — Какви ще бъдат критериите за подбор на участниците в изследването?

— Засега можем да кажем, че ще се стремим да включим училища както от София, така и от други градове. Защото разлика в социалната среда в големия и малкия град безспорно има. Ще се изследва по един клас от няколко училища.

КВ: — Кога ще започне практическата работа по програмата? Какви срокове предвиждате за окончателното й завършване?

— Тази година е подготвителна, самото изследване ще започне през 1990 г. Минималният срок, който трябва да бъде приет от всички участници в изследването, е едногодишен. Разбира се, добре би било да се продължи и по-нататък. В зависимост от богатството на резултатите ще проличи практическата приложимост на изследванията. Ние очакваме, че след приключването на програмата ще имаме нови познания, а вероятно и изводи за това, как ще се обединят училището и новата технология.

КВ: — Какви специалисти ще участват в международната програма, кой ще я ръководи?

— Ще се съставят смесени научни екипи от педагози, психолози, филолози, социолози, специалисти по конструиране на компютризирана учебна среда. Всяка страна има свой изследователски екип, като координацията между всички е възложена на българския. Той работи към Програмата за изследване на човека и неговия мозък.

КВ: — Предвижда ли се да се изследва и физическото състояние на децата, обучавани в компютризирана среда?

— Физическото състояние не е в центъра на вниманието на това изследване. У нас вече се проведоха проучвания върху такива фактори като умора, различни физиологически параметри, които показаха, че при нормално дозиране (около 2,5 часа на ден за децата) на работата с компютър вредни физиологични последици няма. При днешните системи на обучение това време е много помалко.

КВ: — Някои родители се тревожат от увлечението на децата си по компютъра. Имат ли основания за подобна тревога?

— У всеки човек с възрастта се развива твърде типичната черта, наречена консерватизъм. Затова той посреща с недоверие всяко ново нещо. Това може да обясни една част от реакциите на онези родители, които не познават добре възможностите на компютъра. Основания за тревога няма. Проблемът е, че в хода на все по-компютризиращото се обучение за родителите ще става все по-трудно да общуват с децата си.

Аз мисля, че те се страхуват точно от това, а не от недоказаната вреда за здравето. Много родители разбират, че децата усвояват бързо и лесно една нова технология, нов начин на мислене, които по свое време не са могли да усвоят. Те се страхуват може би от изпреварването, което получават децата им и което може да затрудни техните взаимни контакти.

КВ: — Вашата рецепта за такива родители?

— Много наблюдения вече показват, че е полезно в семейството да се обсъждат проблемите на детето, което се връща от училище с новата информация. Такъв начин на общуване би премахнал психологическите бариери, появили се след внедряването на компютъра като средство за обучение.

КВ: — Мислите ли, че новата образователна технология, която се появява вследствие от въвеждането на компютъра, може да направи революция в просветната система?

— Отделните технически средства не правят революции. Тяхната социална употреба обаче може да доведе до социални изменения.

Засега все още е рано за подобно предвиждане, защото тази технология и свързаната с нея педагогическа методология могат да тръгнат в различни посоки, които в този момент не са ясно очертани. Едно е сигурно — през следващите 10—15 години всичко това ще стане съставна част от педагогическия процес — само по себе си вече достатъчно дълбоко изменение. У нас сега се правят първите крачки, като всяко начало съвсем скромни по мащабите си. В много други страни новата педагогика се е превърнала в част от учебния процес. Смятам, че България върви уверено по същия път.

Интервюто взе
инж. ТАТЯНА СВИЛЧЕВА

ВЧЕРА

ПРАДЯДОТО НА КОМПЮТЪРА

(Щрихи от портрета на Чарлз Бабидж)

ЧАВДАР СЛАВОВ

Втората половина на XIX век наистина подари на човечеството огромно количество важни изобретения — локомотива на Стивънсън, парахода на Фултон, безопасната миньорска лампа на Дейви и телеграфа на Морз. Това е период обаче не само на радикални технически преобразования, но и на крупни научни открития. При това много учени от тази епоха, виждайки пред себе си огромно поле за творчество, се втурнали в няколко области едновременно. Такава е биографията на Гаус, Ампер, Ерстед, Карно, Доплер и на много други.

Чарлз Бабидж (Charles Babbage), роден в Девоншир през 1772 г., станал член на Лондонското кралско научно дружество на 24 години. Началото на XIX век той посрещнал с трактати по физика, геология, астрономия, математика. Неговата разностранност била удивителна. За девиз избрал принципа „Истината преди всичко“. Той се ориентирал добре в археологията, топлотехниката, часовникарството, но най-блестящи резултати постигнал в теорията на функциите и на диференциалното смятане. В тази област придобил славата на забележителен математик.

В Лондон Бабидж споделя със свой колега, че смята за нетърпимо положението математици, астрономи и физици да губят ценното си време в рутинни изчисления. На него му се искало по-бързо да се приближава до истината, до същността на сложни въпроси и да посвещава усилията си на решаването на проблемни задачи. „Нужна ни е машина за смятане“ — и това не е било само мечта, а и формулиране на задачата, която поставил пред себе си.

Действително математиката от миналия век била затлачена от рутинни пресмятания. И Бабидж избрал правилното направление да се преодолее тази пречка. Той решил да създаде „диференциален двигател“ — използвал термина „двигател“ в смисъла на „система от механизми“. И през 1819 г. разработил принципите, заложени в основите на автоматичните сметачни машини. Правилно преценил, че всяко изчисление може да се разложи на редица елементарни операции, които могат да бъдат изпълнени от механизми. Изоставяйки изцяло науката, Бабидж се посветил всеотдайно на тази идея. По неговия замисъл машината трябвало да решава задачи като сортиране на данни и сравняване и проверка на сложни съчетания от цифри, предложени ѝ от човека. Както може читателят да се убеди и сам, тази идея е изпреварила времето си с повече от 100 години — първите сполучливи електро-механични калкулатори са били създадени едва в 1938 г.

В 1820 г., получавайки субсидия от 17 000 лири, Бабидж започва да конструира своя двигател. Той обаче много бързо ги похарчил заедно с личните си спестявания от 6000 лири, след което се убедил, че идеята му е неизпълнима. Технологията на епохата можела да му предложи само много тромави и несигурни зъбни предавки. Но след кратък отдих Бабидж преодолял своята неудовлетвореност и с нова енергия започнал да разработва втори вариант — този път — на „аналитична машина“. Това вече било по-сложен калкулатор, който трябвало не само да събира и умножава, но и да решава уравнения.

Бабидж предлагал машина, способна да върши известна творческа работа. Нея с увереност можем да наречем прик предшественик на съвременните електронноизчислителни машини, защото бил проектирал механизъм с памет и блок за сравняване на получените резултати с еталони. Последователността на операциите се задавала с перфокарти (!). Бабидж предложил и печатащо устройство, работещо на принципа на гравиране върху медна пластинка. Естествено за втория вариант Бабидж повишил техническите изисквания към изработването на детайлите. По същество основният блок представлявал голям часовников механизъм с множество зъбни предавки, валове, гърбични и зъбностопорни механизми. По терминологията на Бабидж този блок се състоял от две части — „склад“ и „мелница“. „Складът“ се монтирал върху вертикална колона с 41 секторни зъбни диска. Тяхното разположение се базирало на десетичната бройна система. „Мелницата“ от своя страна се състояла от няколко вертикални колони, които изпълнявали изчисленията. Перфокартите задавали на различните сектори на какъв ъгъл да се завъртат, за да извършат съответната операция.

Бабидж се опитал да заинтересува с калкулатора си банкери, фабриканти, учени, политици, но никой не се отзовал и той не получил повече субсидии отникъде. Тогава през 1831 г. обвинил правителството и Кралското дружество в липса на широта на възгледите, в предубеденост — че не разбират една от главните задачи на века: да се механизират и автоматизират изчисленията. Такава била и действителната картина, гледана от високите на нашето съвремие. Казано съвсем честно — за да се формулира и да се заяви тази цел преди 150 години, е била необходима истинска научна смелост. Но отговор не последвал. Единственият човек, който подкрепял Бабидж, била жена — английската математичка Ада Лъвлейс, дъщерята на Байрон. Тя направила първите в света програми за изчислителна машина. Една от

КВ. 03–04 89

тях представлява пресмятането на числата на Бернули.

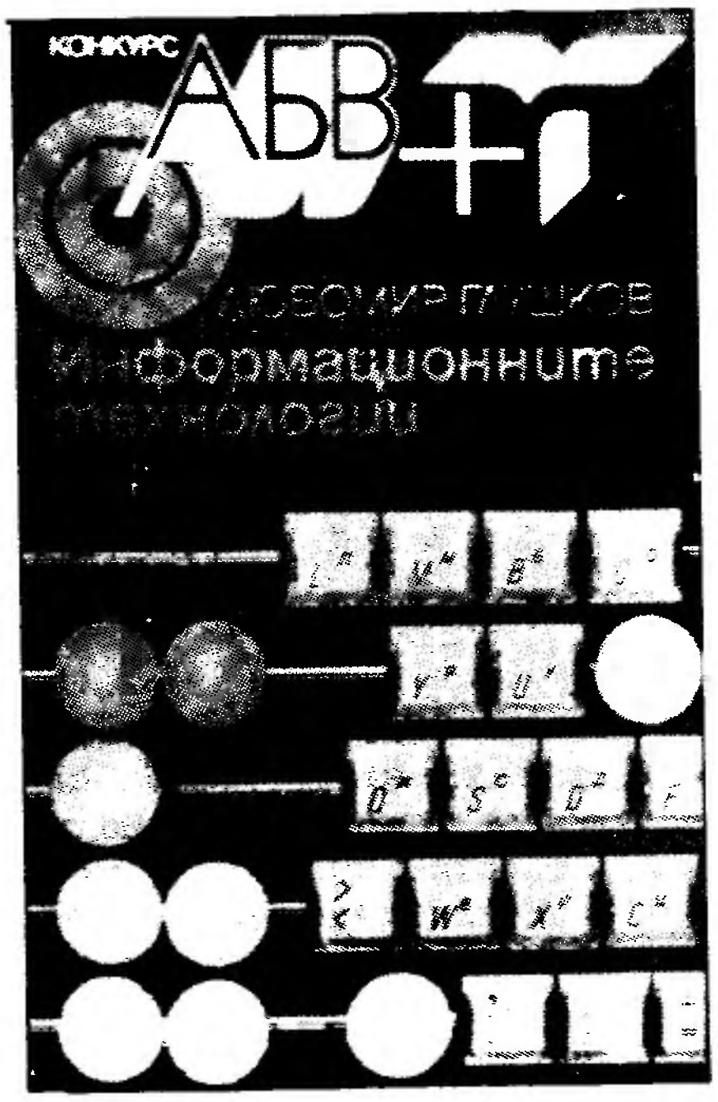
Сблъсквайки се с крайната ограниченост и деспотизма на научните „авторитети“ на своето време, с мракобесието и алчността на властта, Бабидж загубил останалите 37 години от живота си в опити да превъплъти идеята си в метал. Но и тук той бил препънат от значителни трудности. Инженерната технология тогава не могла да му осигури необходимата прецизност при изработката на зъбните колела и гърбиците. Навремето тези трудности се оказали непреодолими. Механиката, залегнала в самата същност на машината, била главната преграда. Зъбните колела се зацепвали зле, неочаквано възниквали места с повишено триене и в крайна сметка някой от възлите блокирал. Между другото инженерите, запознати с трудностите по реализирането на проекта, си направили необходимите изводи и впоследствие положили много усилия, за да издигнат прецизното металообработване, което представлявало действителен принос в английското машиностроене и часовникарската индустрия. Но тогава Бабидж не бил вече между живите.

През 1881 г. на един известен английски математик поръчали да подготви статия за Бабидж, предназначена за Британската енциклопедия. Той се запознал с машината — експонат в Британския музей, и заключил, че подобни машини принадлежат на фантастиката. Че те винаги ще представляват само теоретична възможност за механизиране на изчисленията. Този невнимателен отзив дълго време отпъждал математиците и никой не се осмелявал да повтори опита на Бабидж. Но в края на краищата победили необходимостта и ентусиастите. Калкулаторите навлязоха във всекидневието на хората и днес те се срещат почесто, отколкото това може да се твърди дори за телефона.

А що се отнася до машината на Бабидж, неотдавна група специалисти я прегледали и установили, че с незначително донапасване на дисковете и секторите механичният калкулатор може да бъде пуснат да работи — да пресметне същите числа на Бернули, за които има готова програма от Ада Лъвлейс.

□

РЕЦЕНЗИЯ



„ГОЛЯМАТА КОШНИЦА“ Е ПЪЛНА!

ДОНЧО ХРИСТОВ

Читателю, преди да продължиш с рецензията за книгата*, чиято корица виждаш на снимката, се налага едно малко обяснение. Става дума за необикновената емблема, която придружава заглавието — АБВ+Т. Всъщност, ако си се вгледал по-внимателно и си прочел думичката „конкурс“, набрана с по-дребен шрифт, може би си успял вече да разкодираш какво се крие зад тази формула. Но нека не злоупотребявам повече с търпението ти, ако не си отгатнал „шифъра“. И така това е първата книга от успешно приключилия Първи анонимен конкурс за написване на научнопопулярна книга, организиран от вестник „АБВ“ и издателство „Техника“.

Формулата, както казват, си е формула. Да я напишеш е лесно, но главното е какво стои зад нея. И как се е стигнало до създаването ѝ. В случая това стана, след като от петте предварително представени материала по темата „Информационните технологии“ и въз основа на рецензиите за тях журито — от представители на издателството, на вестника и на ТСО „Българска книга и печат“, награди доц. к. т. н. инж. Любомир Глушков и определи той да напише книгата. Авторът е специалист по приложение на компютрите в управлението, участвал е в изграждането на първите изчислителни центрове у нас и работи в Института за социално управление към АОНСУ, където завежда катедра по технология на управлението. Всъщност всички тези подробности от биографията му станаха ясни по-късно, след като изборът беше направен и бе отворено малкото пликче с данните за автора — напомням, че конкурсът бе анонимен. Тогава именно организаторите си отдъхнаха с известно облекчение — надеждите, че са попаднали на подходящия Автор, изглежда, щяха да се оправдаят. И сега, когато всеки може да купи и оцени книгата — включително и компетентната и взискателна аудитория от читателите на списанието, — спокойно може да се каже, че „голямата кошница“ на надеждите действително е пълна.

Със завидно майсторство авторът е направил дисекция на информационните технологии и е показал тяхното съвременно състояние. Започвайки с кратък преглед на един от основните им клонове — комуникациите, — той увлекателно разказва и за другите два „кита“, на които те се крепят — микроелектрониката и компютъра — главният „виновник“ за събирането, съхраняването и обработката на информацията. Оттук нататък Л. Глушков описва взаимодействието на тези три направления, като се спира на най-типичните и най-ефективните им днешни приложения — текстооб-

⇨

---

* *Любомир Глушков. Информационните технологии. С., Техника, 1988. (Конкурс „АБВ“ + „Т“).*

работка, електронни таблици, компютърна графика, автоматизирано проектиране. Казвам „днешни“, но в книгата убедително присъства и утрешният ден, когато се очаква създаването на пето поколение компютри, на всепроникващ изкуствен интелект и на мощни експертни системи, от които най-оптимистично настроените им създатели очакват не само да обработват знания, но и да ги произвеждат. Този обоснован и сдържано поднесен прогностичен елемент, присъстващ в книгата, е едно от най-поразяващите й качества.

Само благо ли са информационните технологии? Отговор на този въпрос авторът дава накратко в последната глава, където интригуващо се разказва за информационната престъпност, за „вирусите“, които нападат компютъра, за екрана на терминалите, който не само уморява очите, но води и до повишено умствено натоварване. Тук се споменава и за законодателството, което предстои да се създаде, защото възникват казуси без прецеденти, следствията от които засягат по-голямата част от обществото и поставят на изпитание традиционни и привидно непоклатими норми на морала.

В книгата няма формули, чертежи, таблици, графики — неизменният аксесоар на „по-тежките“ книги. И това — не си правя илюзии — навярно ще разочарова почитателите на по-солидните издания. Няма да споря с тях — всеки има свои предпочитания и вкусове за ролята, мястото и същността на научнопопулярната книга. Едно обаче е безспорно — че разглеждайки съвременните информационни технологии и техните приложения, авторът е успял да разкаже достъпно и увлекателно, на много места с живо чувство за хумор, за една съвременна област на знанието така, че и хората без достатъчно предварителни знания и специална подготовка да го разберат. И още нещо, което е не по-малко важно. Авторът майсторски е „вписал“ съвременните информационни технологии в контекста на цялото развитие на човешката култура — хуманитарна и научна, така че дори и непосветените да почувстват тази приемственост. И това е така необходимият вътрешен подтик, който ще накара младия читател да пожелае да се гмурне в дълбините на „информационния океан“ и може би да направи от информационните технологии своя съдба.

□

АКТУАЛНО

ОЩЕ ЗА КОМПЮТЪРНИТЕ ВИРУСИ

Н. с. инж. ВЕСЕЛИН БОНЧЕВ

Темата за компютърните вируси става все по-актуална. Още повече че вълната им достигна и нашата страна... Какво да се прави, прогресът в областта на компютрите си има и своите лоши черти. Списание „Компютър за вас“ своевременно отговори на читателския интерес (виж КВ.04—05.88 и КВ.01—02.89). В тази статия ще споделим някои нови сведения, получени от чуждестранния печат, и ще разкажем по-подробно за проникналите у нас вируси, както и за мерките за борба с тях. Но най-напред

МАЛКО ИСТОРИЯ

Как е възникнал първият вирус? (Отсега нататък под думата „вируси“ ще разбираме компютърните вируси освен в случаите, когато това е специално отбелязано.) Както може да се очаква, това е станало до голяма степен случайно.

През 1972 г. М. Макилрой от „AT&T Bell Laboratories“ изобретил играта „Дарвин“ (1). По правилата на тази игра всеки играч представя известно количество програми, написани на асемблер, които се зареждат в паметта на компютъра заедно с програмите на другите играчи. Тези програми се наричат „организми“. Организмите, създадени от един играч (т. е. принадлежащи към един и същ „вид“), се опитват „да убият“ представителите на другите видове и да заемат „обитаваното“ от тях „жизнено пространство“. За победител се смята този играч, чиито организми в края на играта са повече. Разбира се, тук не става и дума за истински вируси. Но идеята за програми, които „се сражават в тъмните и безшумни дебри на компютъра“, вече витаела във въздуха.

През 1976 г. във фирмата „Хегох“ била създадена системната програма WORM („Червей“). Червеят се зареждал в несъпротивляващия се компютър под контрола на програма-супервайзор. Задачата му била максимално да интензифицира използването на ресурсите на мрежата от компютри на фирмата. За целта той поемал управлението на машината, в която бил зареден, и в сътрудничество с червеите в другите машини осигурявал изпълнението на големи приложни програми в образуваната се по този начин мултипроцесорна система. Червеят бил така устроен, че всеки, който пожелаел да използва една от заетите машини, можел да я получи на свое разположение, без да нарушава работата на системата.

През 1984 г. в една американска фирма (името й не ми е известно) намерила разпространение играта Animal („Животно“), притежаваща почти всички черти на съвременния вирус. Програмата предлага на играещия с нея да си намисли някакво животно и да се опитва да го познае с въпроси (има ли козина, във водата ли живее и т. н.). Ако програмата не успее за отгатне намисленото животно, тя смирено моли играча да й съобщи какъв въпрос трябва да се зададе,

(1) „Software: Practice and Experiance“, vol. 2, 1972. pp. 93—96)

за да разпознава животното в бъдеще. Този въпрос се запомня и така програмата „се самообучава". Тази игра е разпространена и у нас във версия на Бейсик за Правец-82 и на Турбо Паскал за Правец-16. И двете версии обаче нямат „вирулентните" свойства, описани по-долу. Работата е там, че програмистът на споменатата фирма искал да повиши ефективността на играта. Той се възползвал от факта, че използваният от фирмата компютър (вероятно машина от класа на PDP-11) предоставял на всеки потребител собствена директория за работа, но нямал механизъм за защита на директориите от други потребители (или от техните програми). Програмистът променил играта така, че след завършването си (т. е., когато е станала „по-умна" — ако играчът въведе нови въпроси, или във всеки случай „не по-глупава" — когато не ги зададе) програмата се копирала в директорията на още някой потребител. Ако в тази директория вече е имало по-ранно копие на играта, то се изтривало и на негово място се записвало новото. Благодарение на това поведението на програмата ставало доста неочаквано за играещия с нея. Ако пък в директорията дотогава липсвала версия на „Животното", играта се предлагала на още един потребител.

Играта била доста популярна и скоро във всяка директория се съдържало по едно нейно копие. Нещо повече, когато сътрудниците на фирмата се премествали на работа в друг отдел, те вземали със себе си и самата програма. По този начин с течение на времето тя се разпространила върху всички компютри на фирмата.

Ситуацията, общо взето, не била опасна, но многобройните копия на тази сравнително безобидна игра започнали съществено да задръстват дисковото пространство. Изтриването им не било ефнкасно — достатъчно било да се забрави само едно копие и то скоро се размножавало отново.

Въпросът бил окончателно решен чрез средство, което според съвременната терминология може да бъде наречено „антивирус". Между другото, това е характерен пример как с механизма на вируса може да се свърши и полезна работа.

Създадена била версия на програмата, която след завършване на играта се копирала не в една, а в две други директории. Така за кратко време всички стари версии били заменени с „по-жизнеспособните" нови. След една година обаче, след настъпването на определена дата, всичките копия на новата програма променили поведението си. Вместо да се самовъзпроизвежда в два екземпляра при всяко обръщение към нея, сега програмата играела една заключителна партия, казвала на потребителя: „Сбогом", и се самоунищожавала. Така „Животното" било изгонено от системата.

В средата на 80-те години Рут Люорт от Холмдел, щата Ню Джърси, създала своеобразно чудовище дори без да напише истинска програма. Компютърът, на терминала на който тя работела, се управлявал от операционната система UNIX. Рут подготвяла демонстрационна версия на своя обучаваща програма. Изведнъж системата се претоварила и започнала да реагира изключително бавно. Рут с ужас си спомнила, че е указала знака амперсанд („&") като разделител на полетата данни, въвеждани от терминала. Но в ОС UNIX този знак служи за пораждането на фонов паралелен процес! Очевидно, когато компютърът е приел съобщението от терминала, амперсандите са били прехванати и са породили няколко процеса, които от своя страна са породили нови процеси, и така до безкрайност. Наложило се системният администратор да бъде уведомен и компютърът (който на всичкото отгоре се намирал в друг град) да бъде спрян и стартиран отново. Не се знае колко е струвало всичко това на фирмата и на бедната Рут . . .

Всички тези историн били публикувани от А. К. Дюдни — автор на рубриката „Занимателен компютър" на списанието „Scientific American". Задачата им била да служат като фон за статията на Дюдни, в която той описвал изобретената от него игра Core Wars („Бой в паметта") — една изключително интересна игра, за която някога ще разкажем на страниците на списанието. Двама читатели обаче — Р. Черути и М. Морокути от гр. Брешиа, Италня — схванали нещата съвсем иначе . . . Те си поставили задачата да напишат програма, която да притежава свойството да се разпространява от компютър на компютър без знанието на потребителите. Трябва да призная — една наистина предизвикателна задача. Дълго време двамата италианци не могли да измислят как точно трябва да работи вирусът (терминът „вирус" се употребява за първи път от Марко Морокути). Най-после Черути се досетил, че програмата всъщност трябва да заразява дискетите, използвайки паметта на компютъра само като междинна среда.

По това време най-разпространеният и достъпен персонален компютър бил Apple II. Както е известно, почти всички дискети, предназначени за този компютър, съдържат копие от операционната система (DOS 3.3), при това — на фиксирано място. Черути и Морокути решили, че вирусът трябва да представлява лека промяна в операционната система. При всяка операция „запис на сектор" той трябвало да проверява дали съответната дискета е вече „заразена". Ако ли не, вирусът трябвало да внесе аналогично изменение в операционната система, намираща се на дискетата. По този начин той щял да се размножи по всички дискети, поставяни във флопидисковото устройство на дадена машина, след първоначалното зареждане на операционната система, съдържаща вируса. Двамата вандали си помислили, че ако се запише такава операционна система върху една от дискетите, използвани в най-големия магазин за компютри в града, то скоро в Брешиа ще възникне истинска епидемия.

Но що за епидемия ще е това при един такъв безобиден вирус? Не, вирусът трябвало да бъде „злокачествен"! Например след 16 заразявания (броячът се поддържа на самата дискета) програмата трябвало да преформатира дискетата веднага след зареждането на операционната система. „Но тогава — пише Черути, — ние сами се ужасихме от зловещата си идея и решихме не само да се откажем от нейната реализация, но и никому да не казваме нито дума."

За съжаление двамата италианци имали неблагоразумието да напишат подробно писмо на Дюдни, а той от своя страна — да го публикува. Така духът бил изпуснат от бутилката и нищо вече не било в състояние да го върне обратно.

Първият пострадал бил Р. Скрента-младши, ученик от Питсбърг. Той решил да осъществи на практика идеята за вируса. Резултатът бил плачевен или по-точно — потресаващо ефективен. Създаденият

„звяр“ се размножил по всичките му дискети, по дискетите на приятелнте му и дори по тези на учителя по математика! Скрента написал програма, която трябвало да преследва и унищожи вируса, но тя вече не се оказала толкова ефективна, колкото самия вирус.

Така започнала историята. В сегашния момент по света са изолирани над 30 различни вида вируси и техният списък непрекъснато се попълва. Впрочем терминът „вирус“ стана модерен и с него често пъти се означават програми, които не отговарят на това определение. Затова нека да въведем една

КЛАСИФИКАЦИЯ

на вредителските програми. Най-общо, те могат да бъдат разделени на логически бомби, троянски коне и вируси. Да видим какво представлява всяка една от тях.

Логическите бомби са малки части код, който злонамереният програмист вмъква в големи програмни проекти. Целта им обикновено е да осъществят „отмъщение“, като повредят цялата система, ако, да речем, програмистът бъде уволнен. Така например една голяма система за финансово-счетоводна дейност може внезапно да прекрати работата си и да разруши използваните бази данни, ако името на създалия я програмист престане да се появява във фиша със заплатите. Подобен случай имаше във ВМЕИ „В. И. Ленин“ — София, по времето, когато бях студент. Един ден компилаторът на FORTRAN на машините ИЗОТ 0310 беше отказал да работи . . .

Троянските коне са много по-опасни, защото са създадени с цел само да вършат зло. Обикновено те са глупави (за това, което правят, много ум не се иска) и злобни. По същество те претендират, че са нещо полезно (например някаква игра, инструментално средство или пък добре известна и реномирана програма), докато същевременно правят поразии. Най-простата от тях е да изтрият всички файлове от текущата директория. Това може да направи дори начинаещият програмист на Бейсик. С малко повече познания може да се изтрият всички файлове от всичкн директории по диска. Наистина съществуват много средства, които позволяват да се възстановят изтритите файлове, но задачата се усложнява от това, че е много трудно да се възстанови изтрит голям и силно фрагментиран файл особено ако има много такива файлове. Друга възможна беля е да се нулира основната директория — задача лесно изпълнима, защото мястото ѝ е описано в boot-сектора на дискетата. Ако в същото време бъде нулиран и FAT (също може да се открне чрез boot-сектора), почти нищо няма да ви помогне. Единственото спасение е, ако предварнтелно сте запазили копие от FAT някъде другаде по диска и знаете къде точно. Добре известните пакети Norton Utilities и PCtools предлагат именно такава възможност. Намерете някой от тях и го използвайте (аз използвам и двата). За съжаление те записват информация за това, къде се намира резервното копие от FAT, на последния достъпен сектор от диска. А някоя хитра програма може да вземе това предвид и да го изтрие!

Някой особено ловък троянски кон може така да повреди FAT, че на пръв поглед (след изпълнение на командата dir) всичко да изглежда наред. Например може първите 2 Кбайта от всеки файл да са на мястото си, но след това съдържанието на файла да е всъщност част от друг файл . . . Белята няма да бъде забелязана, докато не се стартира програма от типа на CHKDSK, а тогава ще е твърде късно и нищо няма да може да помогне. На компютрите от типа IBM AT се съхранява определена служебна информация (например типът на използвания диск) в т. нар. CMOS RAM. Това е памет, която запазва записаната в нея информация и след изключването на компютъра, защото се захранва от батерия. Ако тази информация бъде внимателно променена, това може да доведе до (неприятно) изненадващи последици.

Описаните дотук видове троянски коне са истинско бедствие за обикновените потребители. Обаче за потребителя, който редовно си прави архивно копие от ценната информация, те са само досадни. Уви, има и още по-ужасни програми, на които никакво архивиране не може да попречи. Те унищожават данните от диска малко по малко — например прочетат случаен сектор от диска, променят няколко байта и след това го запишат обратно. Ако всичко това се прави достатъчно рядко (например два пъти дневно), действието им дълго време остава скрнто и повредените файлове проникват и в архивите — малко са хората, които ще проверят внимателно всеки файл преди архивирането на 20-Мбайтов диск.

Всъщност могат да се направят и още по-големи бели.

По света са известни много троянски коне. Ерик Нюхауз от Англия поддържа списък във вид на файл, който се разпространява безплатно. Файлът се нарича The Dirty Dozen („Мръсната дузина“), защото първоначално е съдържал дванадесет имена на програми. Сега те са над 200; една малка част от тях е показана на таблица 1.

Таблица 1

123JOKE	Разрушава директориите.
ALTCTRL.ARC	Унищожава boot-секторите.
ARC513.EXE	Твърди, че е архивар; всъщност разрушава boot-секторите. Истинската програма-архивар е дълга около 32 Кбайта.
ARC514.COM	Разрушава boot-секторите. В действителност не съществува .COM версия на програмата ARC.
BACKALLY.COM	След няколко месеца повреждa FAT.
BACKTALK	Извършва запис върху случайни сектори от твърдия диск.
BXD. ARC	Извежда предупреждаващо съобщение, след което повреждa FAT.
CDIR.COM	Повреждa FAT.
CHU NKER.EXE	Вероятно съдържа програмна грешка; може да повреди FAT.

COMPRESS.ARC	Твърди, че е програма, безплатно разпространявана от Borland; всъщност разрушава FAT.
DANCERS.BAS	Показва анимация върху екрана, като същевременно нулира FAT.
DEFENDER.ARC	Повреждa съдържанието на CMOS рама и форматира твърдия диск.
DISCACHE.EXE	Вероятно съдържа програмна грешка; може да повреди FAT.
DISKSCAN.EXE	Твърди, че търси повредени сектори върху диска; в действителност пише върху тях.
DMASTER	Разрушава FAT.
DOSKNOWS.EXE	Разрушава FAT.
DPROTECT	Нулира FAT.
EGABTR	Твърди, че е програма за EGA контролери; всъщност изтрива файлове.
ELEVATOR. ARC	Изтрива файлове; може също да форматира твърдия диск.
EMMCACHE	Съдържа програмна грешка — повреждa файлове и разрушава boot-сектори.
FILER.EXE	Понякога изтрива дискове.
FUTURE.BAS	Повреждa FAT и нулира основната директория.
MAP	Написана е от неизвестния Dorn W. Stickle.
	Разрушава FAT.
NOTROJ. COM	Твърди, че защитава от троянски коне; в действителност нулира FAT, а ако я използвате по-дълго време — извършва хардуерно форматиране на твърдия диск.
TIRED	Още една програма, написана от Dorn W. Stickle, която разрушава FAT.
TSRMAP	Показва наличните в паметта резидентни програми и нулира boot-сектора.
PACKDIR	Твърди, че оптимизира твърдите дискове; всъщност повреждa FAT.
PCLOCK	Разрушава FAT.
PCW271xx.ARC	Променена версия на програмата PC-Write, версия 2.71; повреждa FAT. Дълга е 98274 байта — истинската PC-Write 2.71 е дълга 98664.
PKX35B35.EXE	Истинската програма е PKX35A35; тази повреждa FAT.
RCKVIDEO	Показва на екрана анимация на рок-звезда, докато същевременно изтрива файловете от диска.
SCRNSAVE.COM	Изтрива твърдите дискове.
SECRET.BAS	Форматира диска.
SEX-SHOW.ARC	Изтрива всички файлове от текущата директория.
SIDEWAYS.COM	Унищожава boot-сектора.
SUG.ARC	Твърди, че е програма за сваляне на защитата Softguard; всъщност унищожава FAT във всички достъпни дискови устройства.
TOPDOS	Форматира твърдия диск.
VDIR.COM	Унищожава файлове от диска.
VISIWORD.ARC	Унищожава съдържанието на диска.
WARDIAL1.ARC	Повреждa FAT.

Троянските коне не бива да бъдат смесванн с т. нар. програми-шеги. Една от тях извежда няколко усмихнати малки лица, които танцуват по екрана на компютъра. Друга пък съобщава, че в устройство A: е открита вода, „източва“ я, като издава звук на отпушено клозетно казанче, след което „подсушава“ устройството, като го завърта и издава звук на въртяща се центрофуга на пералня.

Троянските коне не бива да се смесват и с програми, които имат лошо проектиран интерфейс с потребителя. Пример за това е програма, която се нарича DIRDEL и която изтрива цялото текущо поддърво на директориите, без да задава някакви въпроси.

И най-сетне, троянските коне не бива да се бъркат с програми, които съдържат грешки. Въпреки че тези грешки понякога могат да доведат до унищожаване на ценна информация.

Компютърните вируси са най-хитрият от описаните тнпове програмн. По определение вирусът е програма, която се разпространява между отделни компютри по файловете на конкретен компютър без знанието на потребителя. Забележете — никъде не се казва, че програмата обезателно трябва да повреждa нещо! Досущ като бактериите, живиещи в тялото ни, вирусите могат да бъдат полезнн или просто безвредни. Важното е да се разпространяват, а това никак не е лесно да се постигне. Нека сега да разгледаме типичната

СТРУКТУРА

на вируса. Очевидно е, че той трябва да се закача по някакъв начин за изпълним код — било то .COM-, или .EXE-файлове, или нещо друго — драйвер на устройство (device driver), овърлей (overlay), библиотека, или дори boot-сектор. В общия случай вирусът се състои от четири частн, като някои от тях могат да отсъстват.

Първо, нма код, който инсталнра вируса в паметта. Това е необходимо, за да се осигури честото изпълнение на вируса, ако заразената част се изпълнява рядко — например програмата в boot-сектора се изпълнява само веднъж — при начално зареждане.

Ако вирусът е заразил нещо, което се изпълнява често (например файла COMMAND.COM), то инсталирането в паметта не е необходимо. Веднъж активиран, вирусът използва някоя системна функция, например „дискова операция“ или „изпълнение на файл“, за да се изкопира върху друг диск или дискета.

Втората част от вируса се грижи да извърши заразяването — да копира вируса върху дискетата, да го закачи към някой изпълним файл или пък да замести с него boot-сектора.

За да зарази някой .COM-файл, вирусът просто се залепя след края му и променя първата инструкция така, че да сочи в началото на собствения му код. След като бъде изпълнен, той я възстановява и ѝ предава управлението, т. е. изпълнява се оригиналната програма.

Заразяването на .EXE-файловете и драйверите на устройства става малко по-сложно, но е напълно осъществимо. Още повече че форматът и на двата вида файлове е подробно описан в ръководствата. Ако пък се заразява boot-сектор, вирусът се грижи да запази оригиналния код на някое друго място (което обикновено се маркира като повреден сектор, за да не бъде изтрито) и при начално зареждане да му предаде управлението.

Третата част на вируса проверява за изпълнението на някакво условие. Това може да бъде настъпването на определена дата или време, или нещо по-сложно — например броя извършени заразявания. Условието може да бъде и съвсем заплетено — да речем, броят свободни байтове, останали на диска.

Целта на забавянето („инкубационния период“), което се реализира при изчакването на конкретното условие, е една — да се даде време на вируса да се размножи достатъчно, преди да бъде забелязан.

Четвъртата част на вируса има вредителски цели. Тя се активира, когато условието, проверявано от третата част, се изпълни. На практика четвъртата част може да прави всичко, което може и троянският кон. С други думи, тя може да е също така извратена, както и умът на създалия я програмист. При това програмист, който знае, че програмата му ще се изпълнява често и без знанието на потребителя.

ЕДНО ОТКЛОНЕНИЕ

Може би се питате защо ви обяснявам по-подробно какво представляват и как функционнрат вирусите или троянските коне. Работата е там, че тези, които са в състояние да напишат вирус, знаят как да го направят. Освен това практически цялата необходима за това информация се намира в достъпните ръководства. Така че играта на криеница е безсмислена. От друга страна, както гласи латинската поговорка — praemonitus, praemunitus (който е предупреден, е въоръжен). Като сте информирани за начина, по който действат вредителските програми, ще можете много по-ефективно да се защитавате от тях.

В предишната си статия на тази тема (*виж КВ.01—02.89*) аз защитавах тезата, че за добрия специалист вирусите не са съществена заплаха. Поддържам това мнение и сега. За съжаление тогава пропуснах да забележа, че огромното мнозинство потребители на компютри съвсем не са специалисти от висока класа. Те именно са сериозно застрашени от вирусите. В подкрепа на това твърдение говори фактът, че два от проникналите в нашата страна вируси се разпространиха по компютрите в София за по-малко от една седмица! Но по-подробно за това ще споменем след малко. А сега нека да разгледаме кои са най-известните

ВИРУСИ ПО СВЕТА

1. Вирусът Брейн. Създаден в Лахор, Индия, той се е размножил почти по целия свят. Вирусът е безвреден (не унищожава данни) и се разпространява само по дискети. Името си дължи на факта, че върху заразената дискета се създава етикет на том с име '(c) Brain'. Този етикет се създава на известно разстояние след последното име в главната директория. Така етикетът става видим едва когато в директорията се създадат още няколко файла.

Вирусът разполага началото си върху boot-сектора. По този начин той получава управлението при извършване на начално зареждане от заразената дискета и остава резидентен в паметта даже преди операционната система. След това вирусът зарежда втората си част и чак тогава предава управлението на оригиналната програма за начално зареждане. Втората част на вируса заема 3 Кбайта, които са маркирани като повредени сектори.

Това не е чак толкова много, като се има предвид, че командата FORMAT на ДОС-16 при наличие на някакъв дефект върху дискетат маркира най-малко 5 Кбайта ка повредени.

Когато е активен в паметта, в русът се разпространява по диск тите при форматирането им. „да излекувате“ заразената диск та, прехвърлете съдържащите върху нея файлове на друго мяст заредете ДОС от „здрава“ диск та и форматирайте „болната След това върнете файловете местата им.

В този вирус има още една хи рост, която не се наблюдава у н кой друг вирус от този тип. Ако опитате да прочетете първия се тор от заразена дискета (на здр вите дискети там се намира прогр мата за начално зареждане, а болните — първата част от вируса чрез някакво средство за физич ски достъп до секторите (наприме Norton Utilities), вирусът разбере това (ако е активен в м мента) и ще ви покаже оригина ното съдържание на сектора, т. не това, което е в действителнос а това, което би трябвало да бъ там.

2. Италианският подскачащ в рус. Той се разпространява подо но на Брейн, с някои малки изкл чения.

Първо, втората част на вирус заема само един кластър, марки ран като повреден. Върху дискет в един кластър се съдържат 2 се тора. Същинската втора част вируса заема първия от тях, а в втория се намира оригиналнат програма за първоначално заре дане.

Второ, вирусът се разпростран ва и върху твърди дискове.

Трето, вирусът може да се ра пространи само върху компютр снабдени с микропроцесори 808 8088. Причината е, че в кода вируса се съдържа инструкция mov cs,ax.

Трябва да се признае, дос странна инструкция! Никой асе блер не желае да я асемблира, тя все пак съществува. За съж ление (или може би за щастие тя не може да се изпълни от ми ропроцесорите 80286 и 80386 (80186 нямам сведения). Това пре на вируса да се разпространя върху компютри от типа на IB AT. Но да не се успокояваме пр бързано — у нас са най-разпрос ранени компютрите Правец-1 които работят с микропроцесо 8088, т. е. те са беззащитни къ инфекцията.

Вирусът е почти безвреден. К

гато се активира, той извежда на екрана подскачащо топче — знакът с код 7 от аски-таблицата. Топчето се появява независимо от режима (текстов, графичен) и контролера (CGA, монохромен; за EGA и VGA нямам сведения), като отскача от границите на екрана и от някои знакове. В началото може би е забавно, но бързо става досадно и започва да пречи. Условието за активиране на вируса е изключително сложно — системният таймер трябва да има конкретна и доста рядко получаваща се стойност; текущо трябва да е дисково устройство, различно от това, от което вирусът се е заредил в паметта; и в момента трябва да се извършва дискова операция. Само тогава топчето се появява на екрана. Върху дискетите (или дисковете!) вирусът се разпространява при извършването на дискова операция с тях.

Лечението от този вирус е сравнително лесно. Просто изпълнете командата SYS на ДОС-16 върху заболелите носители на информация. Малко по-трудно е да се освободи маркираният като повреден кластър. Щом веднъж откриете къде точто се намира той, програмата DT.EXE от пакета Norton Utilities ще ви свърши добра работа. Стартирайте я по следния начин:

dt <болен диск> : /C <_номер на маркирания кластър> -

Здравите дискове могат да бъдат „ваксинирани“. Достатъчно е да запишете единица в байта с отместване 503 (т. е. в петстотин и четвърти байт) от boot-сектора и вирусът ще смята диска за вече заразен.

3. Вирусът, “изяждащ“ boot-сектори. Той е много вреден! Върху твърдите дискове създава голямо количество изгубени кластъри (т. е. кластъри, маркирани като заети, но несъдържащи се в никой) файл) — около 2 Мбайта на ден. С дискетите положението е по-лошо. Понякога началната им пътечка се оказва форматирана, но с осем сектора вместо нормалните девет. На всичкото отгоре секторите са номерирани от две до девет, т. е. boot-секторът липсва. Това прави дискетата неизползваема. За съжаление не разполагам с други сведения за този вирус, освен успокояващото съобщение, че по всяка вероятност у нас все още го няма.

4. Други вируси. По-известни (и по-разпространени) са:

— Лихайският вирус, заразил за два дни повече от 600 диска в Лихайския университет (САЩ). Този вирус заразява файла COMMAND.COM;

— Израелският вирус, който на 13 май 1988 г. (петък!) трябваше да унищожи софтуера на израелските компютри. Сведения за него се появиха и в нашия печат, но останаха непотвърдени. Под съмнение е изобщо съществуването му; твърде вероятно е да става дума само за слух или шега;

— Вирусът SCORES, разпространяващ се по компютрите Макинтош и създаден с рекламна цел;

— Смъртоносният Пакистански вирус, създаден от двама самоуки програмисти в Пакистан;

— Вирусът по компютрите Amiga, който по последни сведения се е разпространил дори и в Москва;

— Да не забравяме нашумелия вирус, който преди няколко месеца се разпространи взривообразно по мрежата Ethernet в САЩ и много, много други.

Както вече споменах, броят им непрекъснато нараства.

Нека сега да видим кои са

ВИРУСИТЕ У НАС

Да, да! Както вече се убедихме *(виж КВ.01—02.89)* вируси вече има и в нашата страна. Първата лястовичка беше в „СИСТЕМХИМ“ — София, а досега (януари 1989 г.) са ми познати шест различни вируса. Ще ви разкажа за тях в хронологичен ред, т. е. в реда, по който са стигнали до мен.

1. Вирусът на „СИСТЕМХИМ“ всъщност е известен на света под името VHP-648. Той се разпространява по .COM-файлове, като увеличава дължината им с 648 байта (оттам и числото в името му, а какво означава буквеното съкращение — не знам). Вирусът запомня първите три байта от заразявания файл и ги подменя с инструкция за безусловен преход към края на файла. После „се залепя“ след този край. По този начин при стартиране на заразения файл управлението се предава на вируса. Той извършва черната си работа — заразява още един файл, след което възстановява оригиналните първи три байта и им предава управлението. Така заразената програма все пак се изпълнява нормално. Тъй като вирусът не знае предварително от кой адрес ще се изпълнява (той може да се закача за различно дълги .COM-файлове), кодът му е напълно преместваем — достъпът до всички необходими за работата му данни се извършва през един от индексните регистри. Единствената непреместваема инструкция — тази, която зарежда началната стойност в индексния регистър — се самонастройва преди всяко заразяване. Стратегията на заразяване също е интересна. Вирусът се опитва да зарази файл от текущата директория на текущото дисково устройство, а ако не намери подходящ, обхожда директориите, описани в променливата PATH от обкръжението. Така той лесно се прехвърля от дискета на твърд диск — в почти всички системи, разполагащи с твърди дискове, в променливата PATH са описани директории от диска.

Характерен е начинът, по който вирусът разпознава заразените вече с него файлове. Както е добре известно, в директорията наред с файловите имена се пази и друга служебна информация, в частност — времето на последния запис във всеки файл. По-точно, там са записани часът (0-23), минутите (0-59) и половината от секундите (0-29). Вирусът установява у заразените файлове съдържанието на последното поле равно на 31, което съответства на 62 секунди — нещо, което не може да възникне по нормален начин. От друга страна, полето „секунди“ не се извежда никога на екрана — нито от командата dir, нито от известните ми програми за работа с файлове, нито даже от Norton Utilities. По този начин „заразата“ остава скрита за обикновения потребител. Всъщност има едно-единствено изключение. Ако полето, съдържащо времето на последния запис във файла, е изцяло нулирано, то при команда dir това време не се извежда на екрана. Ако обаче само броят на секундите е различен от нула, ДОС-16 извежда '0:00'.

Файловете, маркирани като заразени, не се инфектират повторно — това навежда на мисълта за създаването на ваксина. Не се заразяват и файлове, по-малки от 10 и по-големи от 64000 байта. Горната граница е необходима, защото .COM-файловете не бива да превишават 64 Кбайта, а самият вирус е по-дълъг от 0.5 Кбайта. Описаният вирус съвсем не е безвреден! Когато реши да зарази някой файл, той се осведомява за системното време и ако броят на секундите е

⇨

кратен на 8, то избраният файл вместо да бъде заразен, бива повреждан. Повредата се състои в това, че като първа инструкция в него се записва безусловен преход към невалиден адрес. Тази инструкция замества първите 5 байта от файла и той може да се смята за унищожен, тъй като потребителят едва ли знае оригиналното им съдържание.

В София той се разпространи почти мигновено, като след „СИСТЕМХИМ“ оплаквания се получиха от: клуб „Компютър“ към младежкия дом „Лиляна Димитрова“: много лаборатории на ВМЕИ „В. И. Ленин“; Лабораторията по слънчева енергия към Единния център по физика и много други места. За щастие лекарства има, дори много. Обикновено всеки пострадал си ги изработва сам, в случая това не е особено трудно.

Може би най-сполучливото от тях е програмата EF.EXE, разработена от кубинската фирма „Eicisoft“. Тя позволява файловете да бъдат: претърсванн за наличието на вируса, лекувани и дори ваксинирани срещу него! Справедливостта изисква да кажем, че тази програма се появи у нас доста преди самия вирус. За съжаление тя остана дълго време незабелязана, може би поради факта, че извежданите от нея съобщения са на испански.

Вирусът VHP-648 притежава още няколко особености. Първо, той няма процедура за обработка на критичиите грешки. Затова, ако се опита да се пренесе на дискета, защитена срещу запис, на екрана се появява добре познатото съобщение Abort, Retry, Ignore?, което, разбира се, е твърде подозрително и може да доведе до разкриването на вируса. Второ, когато търси променливата PATH в обкръжението, вирусът спира при наличието на първия низ, който завършва с „PATH=“. Това навежда на мисълта за една лесна защита без никакви ваксини. Просто вмъкнете командата set vpath=x:\illegal като първи ред от вашия файл AUTOEXEC.BAT. Това ще заблуди вируса да търси файлове за заразяване на място, което по принцип не съществува. Третата особеност на вируса е по-неприятна. Той е асемблиран с някакъв неизвестен (и доста калпав) асемблер. Когато го дизасемблирах, за да го проуча, и след това го асемблирах отново, получи се (в резултат на някои оптимизации, които асемблерът MASM извършва) нов, напълно работоспособен вирус, но с 20 байта по-къс от оригиналния. За съжаление наличните ваксини не могат да се справят с тази „мутация“.

2. Вторият, най-разпространен у нас вирус, е Италианският подскачащ вирус. За него вече говорихме, затова няма да се спирам върху начина му на действие. В София той е попаднал от Русе, където пък е проникнал от Съветския съюз. Там се е появил по време на една школа с международно участие, където е имало няколко компютъра на свободен достъп. По-нататък следите му се губят, но вероятно водят към Лондон. Сега имам достоверни сведения за разпространението му във ВМЕИ „В. И. Ленин“, една външнотърговска организация и клуб „Компютър“ към младежкия дом „Лиляна Димитрова“.

3. Третият вирус, с който се запознах, ми беше предаден от един чуждестранен студент от Гърция. „Вирусът“ представлява програма на Бейсик (!), която, след като бъде изпълнена, генерира файла VIRUS.COM, който е същинският вирус. Той е сравнително безвреден и очевидно е създаден с рекламна цел. При стартирането си вирусът заразява всички .COM-файлове от текущата директория. Заразените файлове се разпознават по думата 'VI' (написана в обратен ред, т. е. 'IV'), която се появява към края им. Характерно за този вирус е, че той „мутира“. Първите му пет поколения се грижат само да заразяват. Едва шестото показва „обратната си страна“. Тя се състои в следното. При стартиране на заразен файл последният, освен че се грижи да зарази и други, с вероятност 0.5 (т. е., ако младшият бит на системния таймер е равен на единица) извежда съобщението

Program sick error:Call doctor or buy PIXEL. for cure description

(„Програмата е болна, повикайте лекар или си купете програмата PIXEL, за да я излекувате“) и завършва, а с вероятност 0.5 изпълнява нормалната си дейност. Не ми е известно дали този вирус е намерил разпространение някъде у нас; във всеки случай старая се от мен да не се получи „изтичане“.

4. Четвъртият вирус, под названието CANCER („Рак“), получих от същия източник и в същия вид — програма на Бейсик, която го създава. По структура той е почти същият като предишния; очевидно е създаден от същия автор. Заразените с него .COM-файлове (и той работи само върху такива файлове,и то от текущата директория) удвояват размера си. При това не се извършва проверка, дали файлът е вече заразен. Така файловете се удвояват по обем при всяко стартиране на някой от тях, докато „изядат“ цялото свободно дисково пространство. За последните два вируса разполагам с лекарство, което съм готов да предоставя на всеки нуждаещ се.

От описаните дотук вируси нито един не се разпространява по .EXE-файлове. Това ме наведе на мисълта, че последното е изключително трудно. За съжаление имах неблагоразумието да изкажа това твърдение публично. Предизвикателството веднага бе прието и резултатът не закъсня.

5. Петият вирус, за който ще ви разкажа, беше създаден у нас. Нарекох го условно VT-88 по инициалите на създателите и годината на създаването му. Той е напълно безвреден — единственото му действие е, че заразените с него файлове свирват леко, преди да започнат работа.

Вирусът заразява както .COM-, така и .EXE-файлове, но не го прави съвсем коректно. На практика той превръща .EXE-файловете в .COM, без да променя имената им. Работата е там, че разширението на файла няма особено значение за ДОС-16. EXE-файловете се разпознават по това, че съдържат буквите 'MZ', стоящи в първите два байта от файла. Всички файлове, които не притежават този идентификатор, се зареждат в паметта, както и .COM-файловете, независимо от разширението им.

Разбира се, тази стратегия на вируса означава, че .EXE-файловете, по-големи от 64 Кбайта, не могат да бъдат заразявани. И точно така става в действителност — вирусът проверява дължината на файла и ако тя не отговаря на горното условие, не го заразява.

Съществуват няколко негови версии („щамове“). Характерно за тях е, че ако по-нова версия на вируса срещне файл, заразен с по-стара, тя премахва по-старата и се настанява на нейно място.

За този вирус разполагам с ваксина, разработена от собствените му създатели.

6. Шестият вирус, известен под името YAN-SHORT, е създаден като интелектуално упражнение. Заразява само .EXE-файлове, но затова пък — произволно дълги. Също е безвреден, но е много „по-музикален“ от предишния — заразените с него файлове при стартирането си свирят мелодията „Янки дудъл“, преди да започнат работа.

Този е единствен от известните ми вируси, за който не разполагам с лекарство, нито с ваксина. Обаче не е особено трудно такива да бъдат разработени.

7. В последния момент научих за един нов вирус, появил се във ВМЕИ „В. И. Ленин“ и заразяващ компютрите VAX. Действието му се състои в това да понижава скоростта на обмен в терминалите до 3 бита в секунда. Засега нямам други сведения за него.

При наличие на толкова много вируси по света и у нас логично е да се запитаме с каква

ЗАЩИТА

разполагаме срещу тях.

По света съществуват множество „универсални“ ваксини, но нито една от тях не е толкова универсална, колкото се твърди в рекламата ѝ. У нас нещата, както обикновено, са на принципа „Оправяй се сам“. В някой от следващите броеве ще разкажа подробно как трабвя да се направи ваксина с широк спектър на действие, която да защитава срещу всички известни ми типове вируси и троянски коне.

Всъщност, когато споменах за защита, имах предвид не само програмна, но и юридическа. Както е известно, в щата Тексас, САЩ, беше приет закон срещу създателите на вредителски програми. Впрочем тук има някои тънки моменти.

На първо място, кой трябва да бъде преследван? Създателите на вируса? Но те често остават неизвестни. Да не говорим, че е възможно те да са създали безопасен вирус, който след това да е бил модифициран от някой злоумишленик. И изобщо това е все едно да се преследват производителите на кухненски ножове за това, че с тези ножове понякога се извършват убийства. Ами ако на всичко отгоре създателите на вируса живеят извън страната, в която е в сила законът срещу тях?!

Тогава може би трябва да се преследват разпространителите на вируса? Но те най-често вършат това неволно: пък и тук е изключително трудно да се докаже умисъл.

Или пък трябва да се наказва системният администратор, отговарящ за унищожените данни? Ако е така, той е достоен за съжаление.

Твърде вероятно е въпросите ми да се сторят глупави на професионалния юрист. И нищо чудно — знанията ми по юриспруденция дори не могат да се нарекат дилетантски. Но много бих искал някой специалист да отговори умно на глупавите ми въпроси.

Засега вирусите все още са сравнително редки и екзотични. Много по-вероятно е ценните файлове от някоя дискета да бъдат унищожени от погрешното изпълнение на командата dir *.* , отколкото от вирус. Все още вирусите предизвикват по-скоро усмивка и се смятат за хитър програмистки трик или за средство да направиш мръсно на бившия си началник. Но неизбежно ще дойде ден в нашето все по-компютризирано общество, когато ще се осъзнае, че те всъщност са оръжие. Уви, това ще бъде съпроводено от твърде печални последици. Защото тогава вирусите ще започнат да се появяват на все по-жизненоважни места.

Как ще се почувствате, ако например изведнъж се окаже, че сумата в спестовната ви книжка е нулирана? Или ако подадете документи за пътуване в чужбина, а ви бъде отказано под предлог, че в момента се намирате в затвора и при това с доживотна присъда? Или ако сметката ви за телефона от последния месец възлиза на 5863 лева и 97 стотинки? Впрочем, за да се случи последното, май няма нужда от вируси . . . Ами ако след поредната кандидатстудентска кампания анонимен „доброжелател“ съобщи на ректора, че в програмата за класация на кандидатите е проникнал вирус? Това е все едно по време на изпита някой да каже, че в една от залите има бомба. Изпитът ще трябва да бъде анулиран. Най-печалното е, че дори не е необходимо съобщението да отговаря на истината . . .

Все пак

МОЯТА ПРОГНОЗА

е по-скоро оптимистична. Предполагам, че съдбата на вирусите ще наподобява съдбата на програмните защити. Най-вероятно в следващите няколко години ще наблюдаваме ескалация от все по-хитри и по-опасни вируси и все по-универсални ваксини. После ще настъпи внезапен спад — когато обществото осъзнае, че не се нуждае нито от едните, нито от другите.

И накрая, няколко думи в заключение.

Готов съм да окажа

ВСЕСТРАННА ПОДКРЕПА

на читателите на списанието в борбата им срещу компютърните вируси. В редакцията ще се намира дискета с ваксини срещу известните ми вируси.

Аз самият съм готов да помогна при почистването на заразените дискове, независимо на какви организации или частни лица принадлежат, стига да са в рамките на София. Досега съм излекувал повече от един гигабайт дисково пространство.

Същевременно ще съм благодарен, ако ми изпращате открити от вас нови вируси, ваксини или статии и материали за тях. Гарантирам връщането на дискетите, ако са придружени от надписан и облепен с необходимите марки плик.

Адресът ми е:
1404 София
кв. „Емил Марков“
бл. 26, вх. „В“, ет. V, ап. 52
Телефоните ми са:
домашен: 58-62-61, от 20 до 22 часа
служебен: 71-401, вътр. 255
Моля ви само, не злоупотребявайте!

□

СТРАТЕГИЯ

ИЗЧИСЛИТЕЛНИЯТ ПОДХОД В НАУКАТА И ПАРАЛЕЛНИТЕ КОМПЮТРИ

К. ф. н. АНИ ПРОЙКОВА

Компютрите често се използуват за решаване на научни задачи. Това им приложение е част от един нов подход, наречен изчислителен. Развит като самостоятелна методология през последните 50 години, изчислителният подход заема отделно място от теорията и експеримента. По своята същност той е симулация на теоретични модели с помощта на компютри. Симулациите позволяват подробно да се изучат микроскопичните свойства на обектите и макроскопичните им следствия в множество случаи, които са твърде сложни за теоретичен анализ и недостъпни за експеримент. Тогава компютрите се превръщат в лаборатории за експериментиране с теории.

Тази нова методология е зависима от развитието на изчислителната техника и с появата на относително евтини и мощни компютри тя бързо се разпространи. Друг двигател за процъфтяването ѝ е разочарованието сред теоретиците, които предложиха красиви фундаментални модели за природата, но не винаги успяват да разберат съдържащите се в тях нелинейни структури. В зримо бъдеще се очаква моделите да бъдат осмислени, а някои от тях и проверени чрез компютърна симулация. Най-обещаващ кандидат за „атакуване“ във физиката е квантовата хромодинамика (КДХ), теорията за силно взаимодействуващи си частици.

Симулациите имат и чисто практични приложения — аеродинамичните тунели за проверка на аеродинамичността на новоконструираните коли се симулнрат, вместо да се строят в действителност. Промяната на условията в тунела и формата на колата се задават като нови входни данни. Получените резултати, а и икономическата изгода в случая задоволяват и теоретици, и експериментатори.

Все повече учени се оказват въвлечени в изчислителната индустрия, търсейки начини не само да получат бърз достъп до новото поколение компютри, а и да повлияят върху конструкцията нм. Някои дори сами си правят специализирани машини.

Изчислителният подход не е просто „паразит“ върху по-старите си събратя — теоретичния и експерименталния. Той създава нова азбука със свои права. В 1953 г. — ранното детство на компютрите — Метрополис разви метод, съдържащ принципите на работа със случайни числа, за симулиране на системи с много степени на свобода (една точка в пространството има три степени на свобода.) Този метод допринесе много за осъществяване на водородната бомба. Оттогава, в най-тясна връзка с най-мощните компютри на деня, той се използва за изследване на поведението на сложни системи.

Пряк резултат от изчислителния подход е появата на фрактални форми — плод на компютърната графика. **Фракталите** са системи, чието състояние не се описва в рамките на класическия целочислен подход. Изображението им се появява често във филми и дори на изложби, с което тези форми стават все по-популярни. Трябва да се отбележи, че фракталите имат важно физическо приложение при изследване на фазовите преходи. Пример за фазов преход е топенето на леда или изпарението на водата. В първия случай се осъществява преход на твърдо тяло в течност, а във втория — на течност в газ.

Друг резултат на изчислителния подход е създаването на **клетъчните автомати** — моделни системи с разнообразно поведение, подчиняващи се на прости правила. Клетъчният автомат е неразделна система, състояща се от модел на изследвания обект и компютър. С други думи компютърът е конструиран съобразно изградения модел и не може да се използува за други цели. Предимството на такъв автомат пред обикновените многоцелеви компютри е гигантското му бързодействие. Например клетъчен автомат, моделиращ еволюцията на Слънчевата система във времето, „реагира“ след 2 секунди на всяко изменение — в частност поява на метеорит. Резултат от отчитането на този факт в програма, написана за универсален компютър от серията на IBM 370, се получава след около една седмица.

Сега се извършват симулации на процеси от всички области на физиката: моделират се електронни свойства на материалите, прогнозира се времето, чрез модели се получават нови вещества, необходими за фармацевтиката. Навсякъде се създават програми за научни изчисления. За да се получат от тях реалистични резултати обаче, необходими са огромни изчислителни ресурси — мощни компютри и много процесорно време.

Ето защо непрекъснато се работи върху подобряване на съществуващите алгоритми и се търсят нови пътища за описване на някои фундаментални физически процеси. Открит стои въпросът, как да се опише задоволително взаимодействието в многоелектронна система, като се знае, че електроните са „индивидуалисти“ — два електрона никога не са в едно и също състоянне в рамките на една система.

Голяма част от проблемите, решавани в рамките на изчислителния подход, имат характеристики, определящи доколко трябва да расте мощта на компютъра с увеличаване на размера на симулираната система.

Един централен въпрос, възникващ при анализ на явленията, е контролирането на статистическата и систематичната грешка.

Статистическата грешка се определя от броя N на изчислените независими конфигурации при симулиране на даден процес и стойността ѝ е $\sqrt{N}$. За да се намали статистическата грешка два пъти, трябва да се увеличи четири пъти броят на конфигурациите, което означава да се увеличн четири пъти компютърното време. Ако резултатите трябва да се получават за предишното време, необходимо е да се купи компютър с четири пъти по-голямо бързодействие.

В изчислителния подход понятието систематична грешка е сходно с това в експерименталния подход, където отразява възможностите на приборите, с които се провежда даден експеримент. В изчислителния подход големината ѝ се определя от отношението N изч/n, където изч е броят на точките, в които се извършва пресмятането, а n е истинският размер на симулираната система. Ясно е, че в рамките на едно пресмятане грешката е постоянна. Тя се намалява с увеличението на посоченото отношение. В систематичната грешка не влиза грешката от аритметичното закръгляване на числата. За да се намали систематичната грешка два пъти при симулиране на една тримерна система, трябва да се извършат пресмятанията за $8 = 2^3$ пъти повече точки. За съжаление изчислителните трудности растат дори по-бързо от нарастването на броя на точките, тъй като времето за генериране на необходимите конфигурации нараства с размера на системата. Този ефект се нарича **критично забавяне** и води до допълнително четирикратно увеличение на времето. След отчитане на всички фактори се оказва, че намаляването на систематичната грешка наполовина се постига с 32 пъти по-голяма изчислителна мощ.

Изискваната точност за голяма част от задачите, които представляват интерес, има определени граници, с други думи — безсмислено е по-нататъшното ѝ намаление. Но тези граници са твърде далеч от възможностите на съвременните компютри.

Какви са начините за постигане на голяма изчислителна мощ? Следната аналогия (заимствувана от шотландските физици Баулер и Кенуей) илюстрира сполучливо възможностите за избор. Нека вашата работа е да пренасяте товари с каруца, в която е впрегнат един кон. Работата се разраства и трябва да се превозва все по-голям товар, а конят вече не се справя. Съществуват следните възможности.

Първата възможност е да се подхранва конят с все по-калорични храни, стигайки до анаболици, за да се увеличи силата му и той да се справи сам с изтеглянето на каруцата.

Този път отразява концепцията за конструиране на суперкомпютри, в които увеличението на бързодействието (мощта) се постига чрез използуване на най-нови технологии.

Известно е обаче, че специално поддържаните животни се изнежват и гледането им се превръща в тежко бреме. Така развитието на суперкомпютри води до все по-скъпи решения, тъй като увеличението на скоростта е вътрешно ограничено от използуваната технология. За разпространената в момента силициева технология не се очаква увеличение, по-голямо от 10 пъти спрямо сегашното състояние (1988 г.). Новите технологии, основани на галиев арсенид или на високотемпературни свръхпроводници, обещават да дадат по-мощни, но и много по-скъпи компютри. Основното предимство на току-що описания подход е, че със старите си умения коларят ще управлява и специално подхранения кон, т. е. разработените по-рано програми ще работят и на суперкомпютрите.

Втората възможност е да се купят още няколко коня, които да бъдат впрегнати едновременно в каруцата. Това е аналогия на паралелния компютър, чиято скорост е пропорционална на броя на процесорите, от които е съставен. Това е модулно решение на въпроса за скоростта и конструкцията на компютъра позволява да се включат допълнително процесори при големи натоварвания. Такава система е надеждна, защото дори и да се развали някой процесор, останалите продължават работата макар и с намалена скорост. Процесорите, на чиято основа е изграден паралелният компютър, са много по-евтини и прости от процесора на суперкомпютъра. Несъмнено новите технологии ще допринесат за развитието и на паралелните компютри. Най-голямо неудобство на идеологията на паралелизма е, че старите изчислителни програми не могат да работят на новите машини без основни промени.

Идеята за паралелна работа на голям брой изчислителни устройства се е появила още миналия век (Ч. Бабидж) далеч преди измислянето на съвременния компютър. В нашия век (1922 г.) тя е развита в книгата на Ричардсон „Прогноза за времето чрез числено моделиране“ и много напомня съвременното схващане на паралелизма, но в книгата отделният изчислителен елемент е човек, а не компютър. Представена е схема на пространствено разположена верига от изчислители, които получават данни и гранични условия от съседите си, а цялата дейност се координира от един човек (център).

ТИПОВЕ ПАРАЛЕЛЕЗЪМ

Тук е уместен примерът за каруцата, теглена от коне. Коларят с камшика подкарва конете и на ударите му отговарят почти еднакво, всеки допринасяйки за извършване на работата. Аналогична е конструкцията на паралелните компютри от типа SIMD (Single Instruction Multiple Data), което означава Една Инструкция към Много Данни (ЕИМД). В такъв компютър централният процесор изпраща една и съща инструкция едновременно към всички подчинени процесори, всеки от които работи с ограничена автономност върху собствен набор данни.

Не всички проблеми обаче могат да се решат в рамките на такава

⇨

архитектура. Да разгледаме какво става на гишетата в авиокомпаниите. В този случай трябва да се обслужат няколко опашки от пътници, като терминалите работят паралелно. Ако работата се изпълнява в режима SIMD и един от терминалите се забави, всички останали се задържат. Следователно всеки терминал трябва да работи автономно със своя локална програма. Такъв паралелизъм е наречен MIMD (Multiple Instruction Multiple Data — Много инструкции към Много Данни — МИМД). Всеки процесор (терминал) обработва независимо от останалите свои данни със своя програма, докато не стане нужда да се обърне към някой друг. Реализирането на обръщенията изисква комуникационна мрежа например като телефонната.

Последователните компютри, които сега са масово разпространени, изпълняват една след друга машинните инструкции. Ето защо в някои класификации те фигурират като SISD (Single Instruction Single Data). Всяка класификация има ограничена приложимост и трябва внимателно да се използува. Например векторните суперкомпютри CRAY не попадат в нито една от изброените групи (SIMD, MIMD, SISD), тъй като в тях всяка инструкция предизвиква обработка на цял вектор от данни. Въпреки непълнотата си посочената класификация е широко разпространена.

През последните 10 години станахме свидетели на голямо разнообразие на паралелни компютри. Някои са уникални експериментални системи, други са компютри със специализирано приложение, а трети са конструирани така, че вършат работата на универсален компютър само че алгоритмите на програмите, работещи на обикновените (последователни) компютри, трябва да се преработят изцяло или да се създадат нови.

Различните системи, които се продават, засега не са съвместими една с друга. Ето защо е разбираемо проявяваното съпротивление спрямо широкото разпространение на паралелната архитектура в сравнение с възприемането на суперкомпютрите. Трябва да се отбележи, че съществува немалка група потребители на паралелни компютри, в която се обменят информация и опит. И като се има предвид, че някои важни научни проблеми се решават именно с помощта на паралелни компютри, може да се очаква, че „хватката" на суперкомпютрите около световния пазар постепенно ще отслабне. Изключително важно е, че паралелните компютри предлагат постигане на голямо бързодействие с по-малко средства, което е примамливо за всички, чието финансиране е ограничено.

Нека видим как може да се въведе паралелизъм. При изпълнение на всеки аритметичен процес се преминава през няколко отделни етапа, всеки от които може да бъде най-бързо преодолян, ако се използува специфичен за етапа хардуер. След завършване на първия етап от аритметичния процес данните преминават за обработка по-нататък, а освободеният хардуер започва работа върху нова група от данни. Такова изпълнение на аритметичния процес води до припокриване на различните етапи, но има и по-добър начин за използуване на хардуера: една инструкция предизвиква преминаване на набор (вектор) от данни през аритметични „тръби", така че едни и същи операции се извършват едновременно над различни данни. Това е идеологията на работа на векторния суперкомпютър. Ефективното му използуване до голяма степен зависи от правилното векторизиране на изчислителната програма. Може и по друг начин да се работи на суперкомпютъра — различни задачи се подават за едновременна обработка. Тук е нужно умело идентифициране на данните, за да се знае на коя задача принадлежат.

Плътността на компонентите, които могат да се интегрират в един чип, е толкова голяма, че често той изпълнява всички функции на компютър. Цената за разработване на успешен (работещ) чип е твърде висока в сравнение с цената на платка, изпълняваща същите функции. Ето защо най-евтин за производство и поддържане ще бъде компютър, състоящ се от голям брой идентични чипове. Ако архитектурата на компютъра позволява да се включват или изключват произволен брой чипове, се получава гъвкава система, чийто размер ще се определя от нуждите и финансовите възможности на потребителя. Засега се търси оптимален отговор на въпроси като: колко процесора трябва да се свържат, как да се разпредели наличната памет, как да се програмира такава машина, че изчислителна та мощ на всички процесори да включи за разрешаване на еди проблем. Резултат на тези търс ния е голямото разнообразие компютри, които решават постав ните въпроси по различен начин

Паралелните архитектури се ра личават по няколко главни при нака. Първият признак е **броят процесорите,** изпълняващи едн програма. Ако той не надвишав десетина, машината се нарич **едрозърнеста;** ако е няколко хиля ди, тя се нарича **дребнозърнеста** Вторият признак е **начинът,** по кой то процесорите изпълняват даден програма. В описания режим SIMD всички процесори изпълня ват по едно и също време една съща инструкция, но върху раз лични данни. В режима MIMD всеки процесор изпълнява своя последователност от инструкции върху собствени данни. Третият признак е **разположението** на паметта. В **системи с разпределена памет** всеки процесор има своя памет, която не е достъпна пряко за другите процесори. Това е най-разпространената конструкция сега, тъй като сумарната памет може да стане много голяма — тя расте с броя на свързаните процесори. В други системи паметта е обща за всички процесори и е възможно да е отделена физически от тях, като достъпът до нея се осъществява чрез комутируема мрежа. При този вариант възниква трудност, когато няколко процесора се обърнат едновременно към една и съща част на паметта.

Доскоро промените в архитектурата на компютрите имаха относително малко значение за потребителите. Причината е, че развитието беше такова, че всички създадени програмни продукти се използуваха и с новото поколение компютри. Програмната съвместимост на машините през този не малък период доведе до създаване на огромни програмни пакети, всеки от които съдържа дългогодишен човешки труд. Тъй като се увеличава бързодействието и паметта на компютрите, очаква се да се разработят бързи алгоритми, предназначени за решаване на трудни проблеми. Производителите на програмни продукти преосмислят стратегията си с навлизането на революционни технологии като VLSI (Very Large Scale Integration) в миниатюризацията.

Развитието на операционни системи за компютри с паралелна архитектура е още в началото. В идеалния случай би трябвало потребителят, без да се интересува от промените в хардуера, да може да прехвърли съществуващите за последователни компютри програми на паралелен компютър. Действителността обаче не е такава. Операционните системи са твърде примитивни, за да изпълнят прехвърляне от този вид. По тази причина усилията са насочени към създаване на интелигентни компилатори, които взимат под внимание паралелизма на компютъра и оптимизират съответно програмата.

Създават се и нови програмни езици. Някои са усъвършенствувани версии на добре познати езици като Фортран и Алгол, в които са включени операции над вектори и матрици, както и възможности за разпределяне на данните между процесорите. Предимство на такъв подход е, че с малки промени програмите за последователни компютри работят и на паралелни. Недостатък е, че разширените версии на езиците не ползуват ефективно паралелизма на машината.

Другият път е развитието на нови езици, например Окам. Те нямат недостатъците на първите, но носят допълнителни трудности — необходимо е да се създават нови програмни пакети и това пречи на широкото им разпространение. Изглежда, ще се наложи смесено прилагане на двата типа езици, за да се използуват максимално качествата на компютъра.

Често се обсъжда въпросът, какво е отношението цена/ефективност за даден компютър. Оценката на това отношение е трудна, защото е невъзможно да се пресметнат фактори като **лесно използуване** и налично програмно осигуряване, които имат пряка връзка с ефективността. Трудно се предлагат смислени критерии за сравнение, защото е ясно, че разликата в скоростта на изпълнение на една добра и на една лоша програма може да стане огромна.

Все пак съществува възможност за примитивно сравнение — въз основа на скоростта на изпълнение на операциите с плаваща точка FLOPS (FLoating-point Operation Per Second). Тази скорост действително има значение за математическите пресмятания, тъй като бързината на тяхното изпълнение зависи главно от скоростта, с която компютърът оперира над реални числа. Доскоро няколко десетки милиона флопса се смяташе за респектираща скорост. Паралелните компютри работят в областта на гигафлопсове (хиляда милиона флопса).

Всъщност сравнението с флопсове се прави от производителите на компютри и най-вече от тези на векторни суперкомпютри. Те обичат да цитират върховите достижения на машините, превишаващи два, а и повече пъти скоростта, достигана в реални приложни програми. Дори когато се твърди, че данните за скоростта са получени при изпълнение на програми, а не на тестови цикли, става дума за специално настроени програми. Като се проследят техническите характеристики на произвежданите компютри, може да се заключи, че неподлежаща на съмнение е единствено скоростта, която производителят е обявил, че не може да се превиши.

Конструирани са паралелни машини, притежаващи до 10 пъти по-голямо отношение цена/ефективност от сегашните векторни суперкомпютри. Например векторният процесор FPS-264 има върхова скорост 38 Мфлопса и струва 1 милион долара. Реалната скорост на изпълнение на една правилно векторизирана програма обаче е около 20 Мфлопса. Най-голямото достижение в областта на векторните суперкомпютри е CRAY X-MP/48, който срещу 20 милиона долара дава средна скорост 800 Мфлопса (това е половината от върховата му скорост). С други думи, за един милион долара се получават 40 Мфлопса.

На световния пазар в момента се продават паралелни компютри FPS T-серия, Intel PSC-VX и Meiko Computer Surface, които са 5 пъти по-бързи, или за 1 милион долара се получават 200 Мфлопса. По-важното е, че паралелните компютри са модулни системи, така че може да се купи система, съобразена с финансовите възможности.

Паралелната обработка предлага нарастване на скоростта над техническите възможности на еднопроцесорен компютър, чието съотношение цена/ефективност в най-добрия случай може да се увеличи 10 пъти в сравнение със сегашните векторни суперкомпютри. В идеалния вариант една програма се изпълнява N пъти по-бързо с N процесора, вместо с 1. На практика обаче е по-бавно.

Активно се работи за създаване на алгоритми, позволяващи да се достигне максимална скорост. Неудобството е, че алгоритмите не могат да се използуват на всяка машина, тъй като са тясно свързани с конкретната реализация на паралелния компютър.

Новите алгоритми трябва да отразяват следните особености:

— разпределение на данните в паметта;

— разпределение на пресмятанията между процесорите;

— комуникация между процесорите;

— връзка между процесорите, ако конфигурацията на системата може да се променя.

Целта е паралелизмът на алгоритъма така да се съгласува с паралелизма на машината, че да се минимизира времето за изпълнение на програмата. Паралелизъм в алгоритъма означава да има независими операции, които се изпълняват едновременно. Този брой се мени в различните етапи на алгоритъма. Паралелизъм в конструкцията означава да има процесори, които работят едновременно. При създаване на паралелен алгоритъм усилията са насочени към достигане на максимална ефективност, определена от $t/(N.t_1)$, където t е времето за изпълнение на програмата от 1 процесор, а t_1 — времето за изпълнението ѝ от N процесора. Ако времето за разпределение на данните е сравнимо с времето за аритметични пресмятания, тогава то е определящо при избор на алгоритъм. В действителност ключов фактор е отношението на изчислителната към комуникационната скорост. Комуникационната техника изостава от изчислителната, а засега много приложения на паралелните компютри са свързани с комуникацията.

Може да се направи основно групиране на алгорнтмите, без да се смята, че то включва в себе си всички съществуващи:

— паралелно изпълнявани независими задачи — множество процесори изпълняват самостоятелно една и съща програма върху свой набор от данни, а един централен процесор събира резултатите;

— геометричен паралелизъм — всеки процесор изпълнява една и съща програма, ползувайки данни от предоставената му област, като обменя резултати с най-близките процесори, които обработват данни от съседни области;

— алгоритмичен паралелнзъм, в който отделният процесор е отговорен за част от алгоритъма и всички данни преминават през всеки процесор.

Алгоритми от първата група се използуват ефективно при анализнране на голямо количество данни. Пример са експериментите във физиката на високите енергии.

Типична ситуация е да се обработват милион събития, всяко от които отнема около 20 s за анализ.

Основна трудност в работата е достигането иа такава скорост при прехвърляне на данни от лента или диск към процесор, че нито един процесор да не чака за прочитане или запис иа данни. При висока скорост на прехвърлянето системата може да постигне увеличение на скоростта на изпълнение, равно на броя на включените процесори. Друг пример е моделирането на процеси, където е нужно да се постигне висока статистическа точност. Големият брой независими конфигурации се изчисляват паралелно.

Геометричният паралелизъм отразява основното за геометрията понятие — разстояние. Геометричното представяне на даден проблем се свежда до разделяне на данните в групи по такъв начин, че данните вътре в коя да е група са по-близки помежду си, отколкото с данните от останалите групи. Казва се, че задачата има геометричен паралелизъм, ако и алгоритъмът включва само локални операции, т. е. операции само върху близки данни.

Например при симулацията на количеството петрол в резервоар се постъпва по следния начин — резервоарът се представя като мрежа с толкова клетки, колкото са процесорите в паралелния компютър и всеки процесор отговаря за това, което става в неговата подобласт. По-голяма част от пресмятанията е свързана с данни, принадлежащи на вътрешността на подобластта, т. е. принадлежащи на локалната памет на процесора. Изключения правят **граничните** данни, които отразяват състоянието в две съседни подобласти и трябва да се обменят между паметите на съответните процесори. Отношението на количеството данни, които трябва да се прехвърлят, към количеството на локално обработваните данни е равно на отношението на повърхността на подобластта към обема ѝ. Тъй като това отношение отразява балансът на комуникация към изчисление е ясно, че трябва така да се подберат подобластите, че да имат минимална повърхност при даден обем.

Комуникационното време зависи и от разстоянието между процесорите и за да не се увеличава, трябва процесорите, обработващи съседни подобласти от данни, да бъдат също съседни или поне максимално близо един до друг. С това се завършва геометричното представяне на проблема, а именно — процесорите да имат геометрично разположение, възможно най-близкото до симулирания обект. Например съвсем лесно е тримерни задачи да се представят геометрично върху квадратен масив от процесори, предоставяйки на всеки процесор да обработва по един стълб от данни. И така, при геометричния паралелизъм една физическа система се симулира посредством хомогенен масив от процесори, като два съседни процесора се синхронизират при прехвърляне на гранични данни. Тъй като всеки процесор извършва приблизително едно и също количество работа, това превръща конструкцията MIMD в нещо близко до конструкция SIMD. Независимо как ще се нарече масивът от процесори (MIMD или SIMD), това е високоефективна конструкция за работа с големи задачи и е относително проста за програмиране поради хомогенността си.

Трудности се появяват, ако алгоритъмът изисква пълна информация за работата си, тъй като се налагат чести прекъсвания в локалната работа на процесорите, за да се обменят съобщения между процесорите. Резултатът е разваляне на геометричното представяне.

Много проблеми от компютърната симулация на научни и инженерии системи се решават чрез геометрично представяне — прогноза за времето, аеродинамични тунели, анализ на структури по метода на крайнте елементи, полупроводници, обработка на образи и взанмодействие на елементарнн частици. Най-проста, а същевременно най-фундамеитална реализация на геометричния паралелизъм е клетъчният автомат, изобретен от Джон фон Нойман през 1950 г., докато търсил начин да направи самовъзпроизвеждаща се машина. Той форулирал задачата в рамките на равномерно клетъчно пространство, т. е. пространство, изпълнено с клетки, всяка от които може да съществува в краен брой състояния. Всяка клетка се развива във времето през равномерни интервали (стъпки) по правила, зависещи само от състоянието на околните клетки. Той успява да докаже, че ако всяка клетка може да съществува в едно от 29 състояния и има четири взаимно ортогонални прилежащи клетки, то съществува конфигурация от 200 Кклетки (1 К = 1024), която съдържа универсалния конструктор. Впоследствие са изобретени много едно-, дву- и тримерни игри на основата на клетъчните автомати. Като че ли няма ограничения за приложението на идеите на клетъчните автомати — от самовъзпроизвеждащи се подвижни автомати до шахмат, от обработка на образи от скенери и томографи до самообучаващи се машини.

Клетъчните автомати са модели на паралелни компютри с геометрично представяне.

Последният тип алгоритмичен паралелизъм се свежда до организиране на мрежа от процесори, всеки със своя определена роля, през които преминават всички данни, както става в заводска поточна линия. Обикновено всички данни се държат в паметта на един процесор, който управлява останалите. Той подава данните на подчинените процесори, които се

нуждаят от по-малко памет за дейността си.

Съществуват трудности, свързани с алгоритмичния паралелизъм. Едната е, че през различните етапи на пресмятане могат да се прилагат различни алгоритми и мрежата, оптимизирана за един алгоритъм, се оказва неподходяща за друг. Например един алгоритъм се използува за сортиране на данни от експеримент, а друг — за анализирането им. Този проблем се премахва, ако мрежата е динамично преконфигурируема. В противен случай данните трябва да се анализират от друга група процесори.

Друга трудност за решаване е как да се изпращат команди до всеки процесор — например за да се подготви за изчисление. Понеже всеки подчинен процесор извършва своя работа, той очаква да получи „лична" инструкция. Възможен начин за разрешаване на проблема е пакетно разпределение на команди и данни по мрежата. Всеки пакет има индикатор, който показва предназначението му и какъв обем данни пренася. Съдържанието на повечето пакети, идващи от управляващия процесор, се състои от код на инструкцията, която подчиненият трябва да изпълни, и параметри на данните. Подобни пакети се използуват от подчинените процесори, за да съобщят по мрежата какво е състоянието им.

Последната трудност е, че често някой от процесорите поглъща по-голяма част от изчислителното време. Ако процесорът не може да се разпредели между няколко процесора, той определя така нареченото **запушващо време** — време за задържане на изпълнението на цялата програма.

В заключение може да се каже, че изчислителният подход в науката доведе до изобретяването на паралелните компютри — модулни системи с големи възожности за разрастване. А е сигурно, че се изисква многократно увеличение на мощта на компютрите за достигане на качествен скок в разбирането ни за околния свят. Надеждите за него са свързани с паралелните компютри.

□

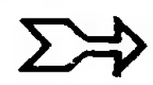
ТЕСТВАХМЕ ЗА ВАС

Първо запознанство с IBM PC DOS 4.0

Инж. ГЕОРГИ БАЛАНСКИ

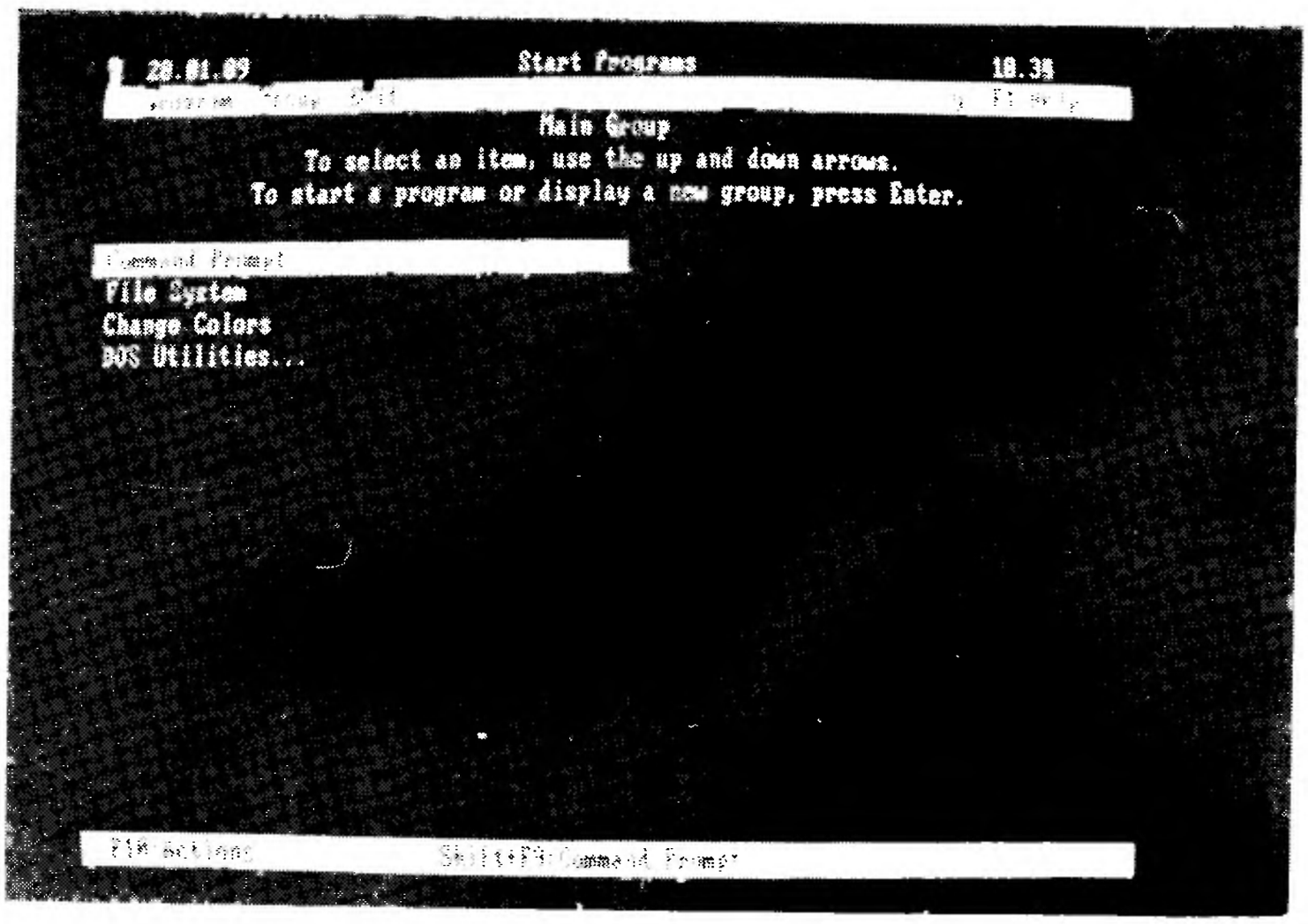
Някак в сянката на разгорещените дискусии за и против толкова нашумялата напоследък операционна система OS/2, създадена за новата генерация PS/2 на персоналните компютри IBM, към края на миналата година без много шум IBM пусна поредната версия на PC \ DOS 4.0, а малко по-късно „Майкрософт" представи и своята версия на този ДОС. Не ще и дума, че появата на нов ДОС е събитие. Докато промените в MS DOS, респективно в PC DOS, извършвани във всяка следваща версия — от първата СП/М-подобна версия през 1981 г. до IBM DOS 3.30, не винаги бяха особено забележими за потребителите — неспециалисти (особено при остарелия хардуер!), то с новата версия вече е направена значителна крачка напред. Ето защо си струва да споделим първите си впечатления от PC DOS 4.0.

Веднага се набива на очи фактът, че новият ДОС се държи значително по-дружелюбно, което е несъмнен жест на производителите към непрофесионалния потребител. Това дружелюбие започва още от подготовката за работа с PC DOS 4.0. При предните версии бе достатъчно ДОС-ът да се зареди от дискетата и с командата SYS C: системните файлове да се копират на твърдия диск. Всъщност това „до-

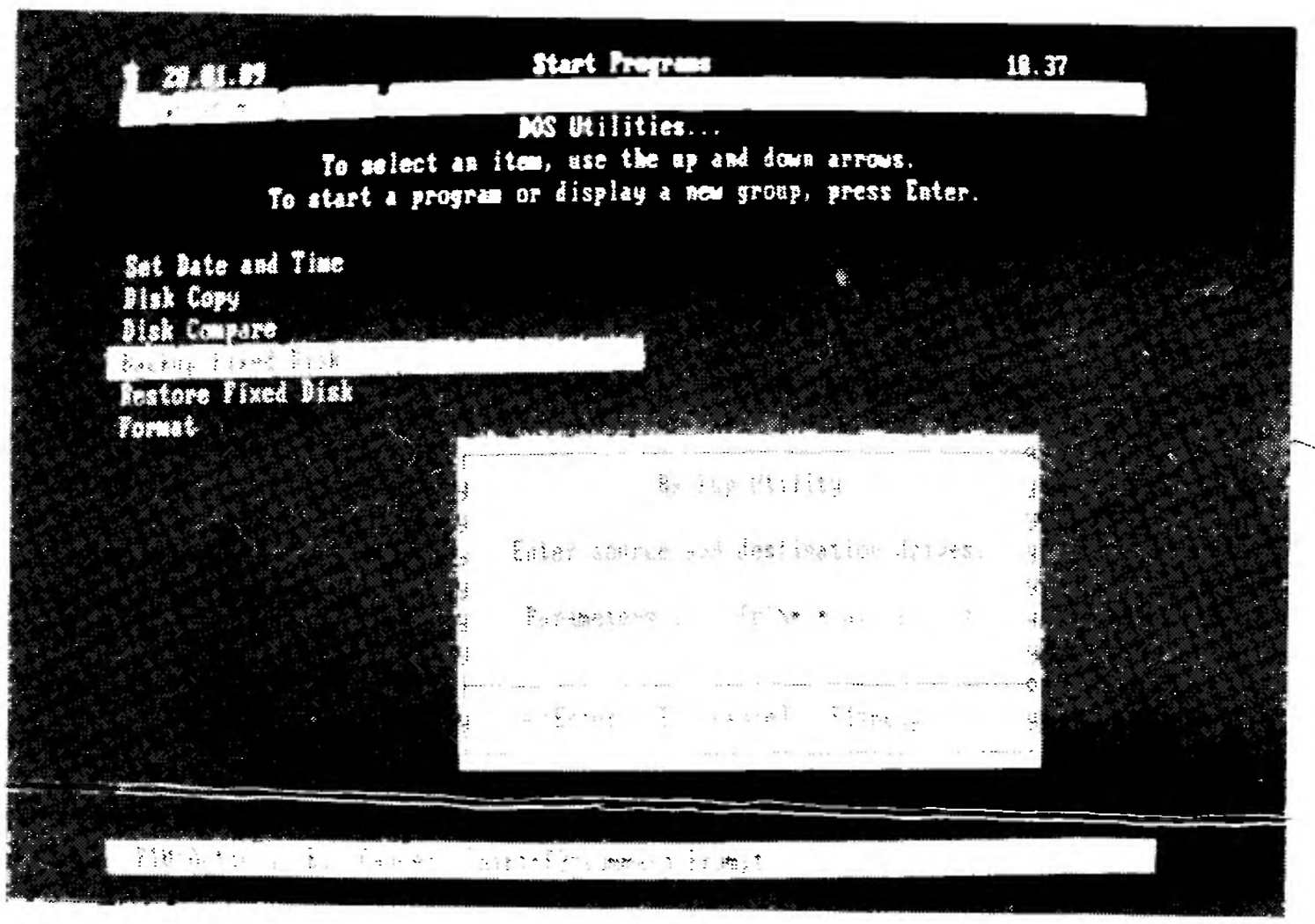


```
Autoexec.BAT

@ECHO OFF
SET COMSPEC=A:\COMMAND.COM
VERIFY OFF
GRAPHICS
VER
KEYB US,,KEYBOARD.SYS

Config.SYS

BREAK=ON
COUNTRY=49,,COUNTRY.SYS
BUFFERS=20
FILES=8
LASTDRIVE=E
SHELL=A:\COMMAND.COM /MSG /P /E:256
DEVICE=ANSI.SYS

Автоматично генерирани
Autoexec.BAT и Config.SYS.
```

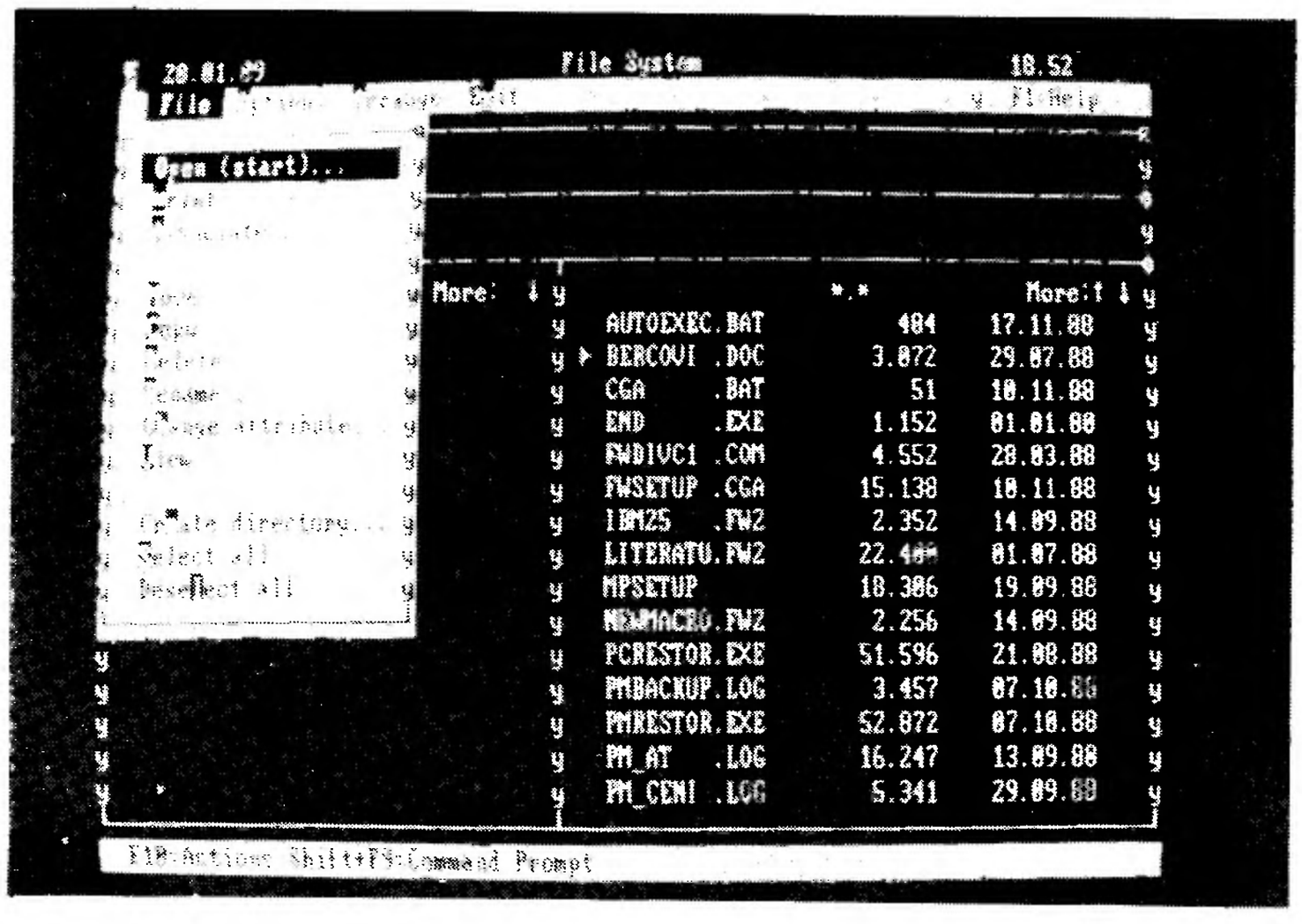
Фиг. 1



Фиг. 2



Фиг. 3



Фиг. 4

статъчно“ е в известна степен условно казано, защото не е чак толкова елементарно да се замени DOS 3.10 с DOS 3.30 поради разликата в обема на системните файлове. Но това е друга тема.

PC DOS 4.0 се разпространява на пет 5 1/4“ дискети (естествено и на 3.5“ дискети), две от които са INSTALL и SELECT. С тяхна помощ и в диалогов режим неподготвен човек може лесно сам да конфигурира ДОС съобразно системата, с която разполага, както и да избере при какви условия ще работи — с максимално използване на възможностите на ДОС и съответно намалена оперативна памет или обратно. Има възможност и за компромисен избор между функциите, които предлага ДОС, и обема на заеманата от него оперативна памет.

Инсталиращата програма запитва за подредбата на клавиатурата, за формата на датата, часа и валутния знак, за типа на принтера, проверява броя и вида на дисковите устройства и т. н. В резултат на всичко това автоматично се създават двата конфигуриращи и подготвящи системата за работа файла CONFIG.SYS и AUTOEXEC.BAT (фиг. 1). Неприятното (за нас!) в случая е, че нито България фигурира в списъка, поддържан от ДОС, нито пък сме виждали някой от принтерите, с които неговите създатели са убедени, че непременно ще се работи! А една от силните страни на ДОС-а е, че поддържа вътрешни драйвери за връзка с периферните устройства.

По описания начин се подготвя инсталирането на PC DOS 4.0 на твърдия диск или на четири 5 1/4“ дискети, ако се работи с флопидискови устройства.

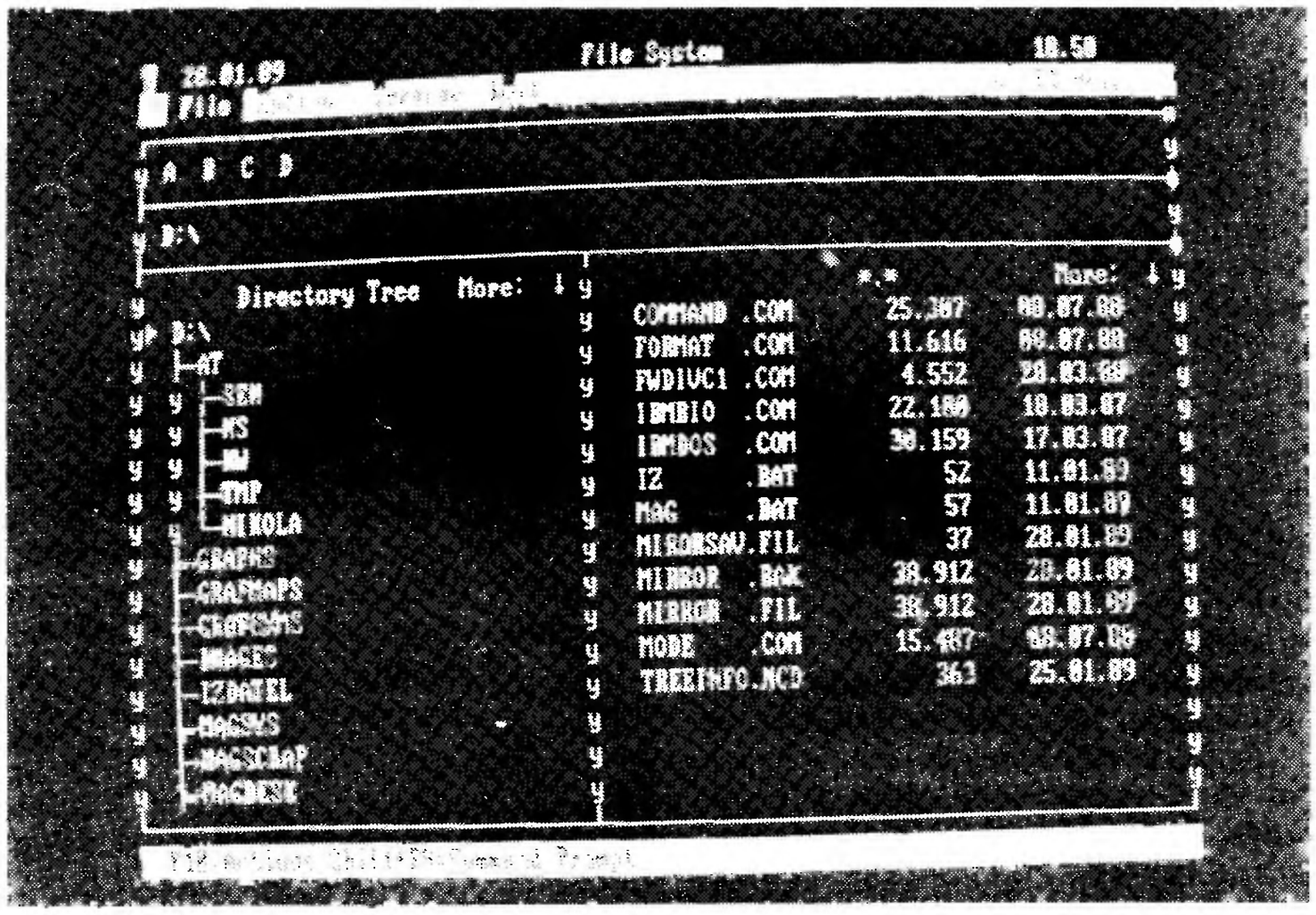
Най-интересната и отличителна новост, значително облекчаваща работата с компютъра, е графичният интерфейс за връзка с потребителя. Нещо, за което плуващите в океана на MS DOS можеха само да завиждат на работещите на Макинтош на фирмата „Епъл“ за удобствата, които предоставя характерният за този компютър графичен интерфейс — множество екрани и менюта, графични символи и пиктограми, които само трябва да бъдат посочени с мишката, за да бъде изпълнена желаната команда.

Тези възможности не бяха съвсем недостъпни и за работещите на персоналните компютри на IBM (и за законните и незаконните им

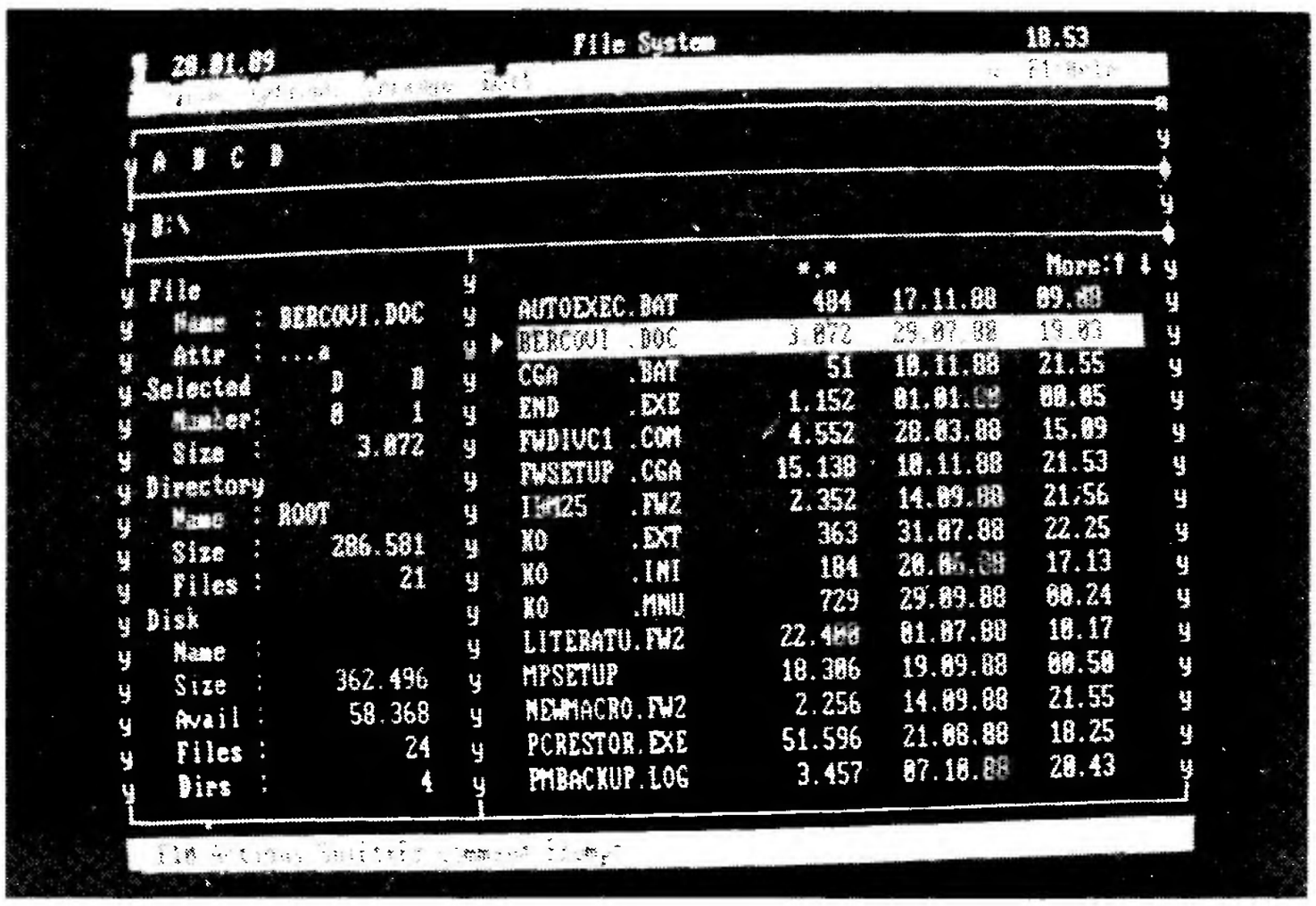
...товчеди), но за целта трябва да се използват различни надстроечни продукти като Windows и МАГ СРЕДА (на СО ППС) или пък помощни програми — популярните „Команден организатор“, Norton Utilities, PC-Tools и други. Всъщност масовото разпространение на тези продукти показва и необходимостта от функционалните възможности, които те предлагат — различни екрани с информация за справочниците и файловете, намиращи се на диска, както и друга помощна информация, менюта за улеснен избор и извършване на различни операции — стартиране, копиране, изтриване, преименуване и преместване на файлове, създаване и промяна на справочниците и т. н. Всичко това може да се прави само чрез избор с курсора, който се управлява било чрез клавиши, или пък с мишка.

Повечето от изброените функции, а и някои други са вградени в PC DOS 4.0. За целта трябва да се стартира като външна команда DOSSHELL. Представа за възможностите на PC DOS 4.0 дават показаните тук екранни изображения. На фиг. 2 е дадена картината, която се изобразява след стартиране на DOSSHELL. Предоставят се възможности за смяна на цветовете на екрана, функции за работа с дискети, функции за работа с файлове и излизане в ДОС — при това от графичната надстройка в оперативната памет остава резидентна част от само 4 Кбайта.

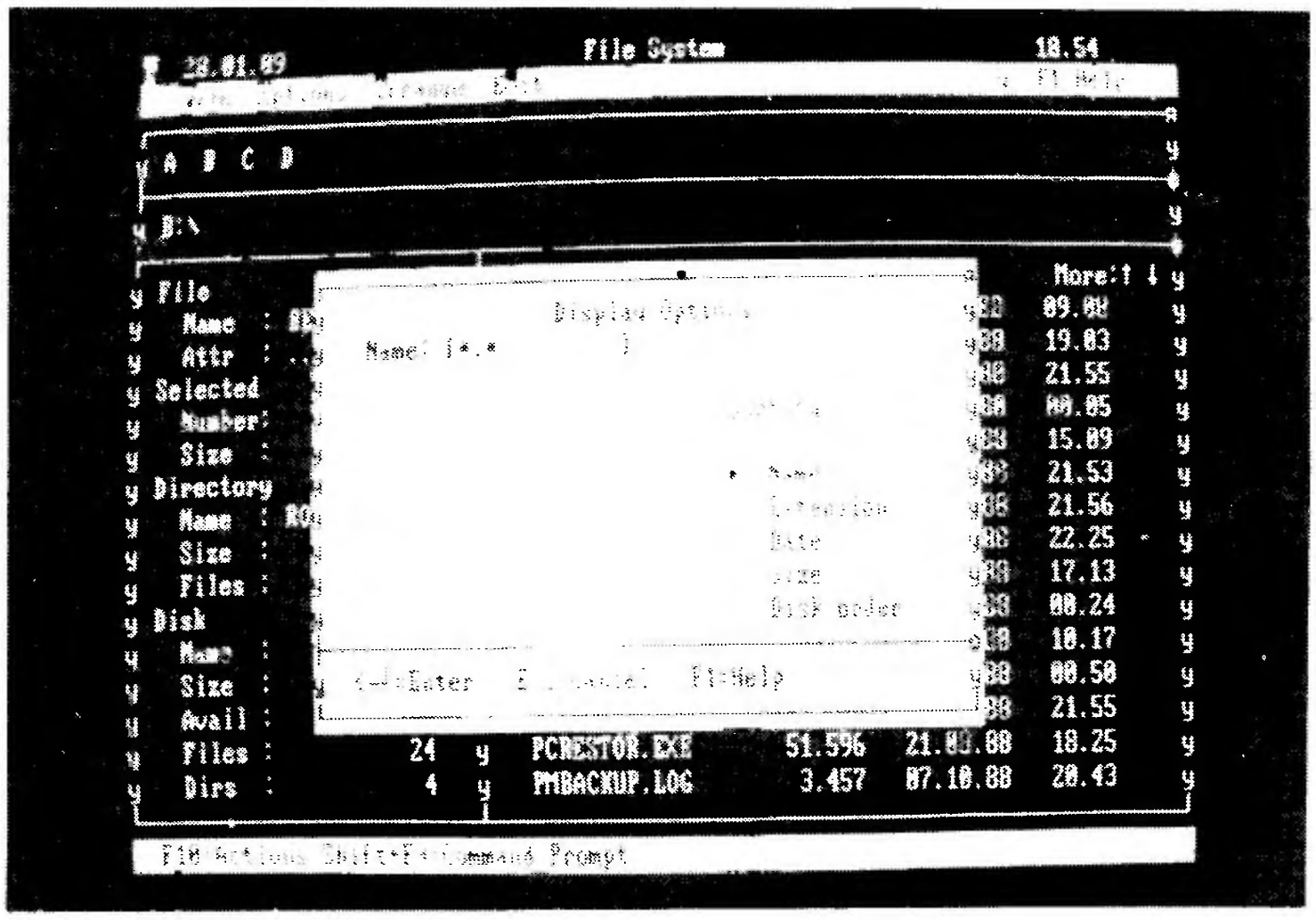
От фиг. 3 се виждат възможностите за работа с дискети — форматиране, копиране, сравняване, правене на резервно копие на информация от твърдия диск и нейното възстановяване. Всъщност се използват познатите външни команди на ДОС, които обаче се въвеждат и стартират чрез командно меню. Това е безспорно улеснение, защото не се изисква точното познаване на съответните команди на ДОС и техния формат. Авторите на PC DOS 4.0 очевидно не са допускали на сериозно, че може да се работи и само с дискетни устройства. В противен случай не биха оставили появата на досадното съобщение Bad command or file name, а биха го заменили с подкана за поставяне на дискетата, на която се намира съответният файл



Фиг. 5



Фиг. 6



Фиг. 7

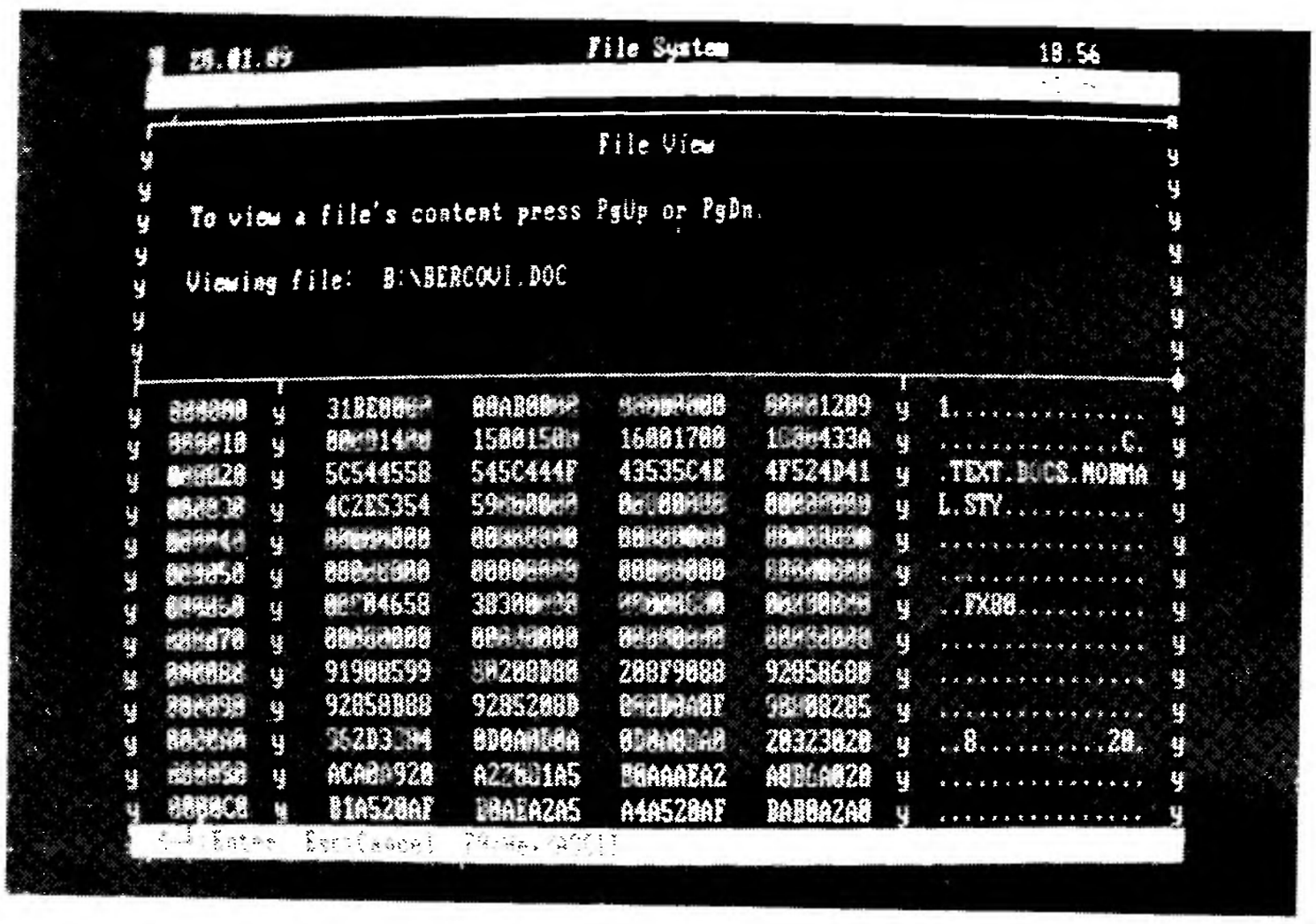
за изпълнение ия външната команда.

Значително повече са възможностите за работа с файлове (фиг. 4). Екранът може да се разделя на две части с информация за две дискови устройства, едновременно да се показват дървото със справочниците и съдържанието на някой от тях (фиг. 5), справочна информация за даден файл, за директорията и диска, на който той се намира (фиг. 6), и т. н.

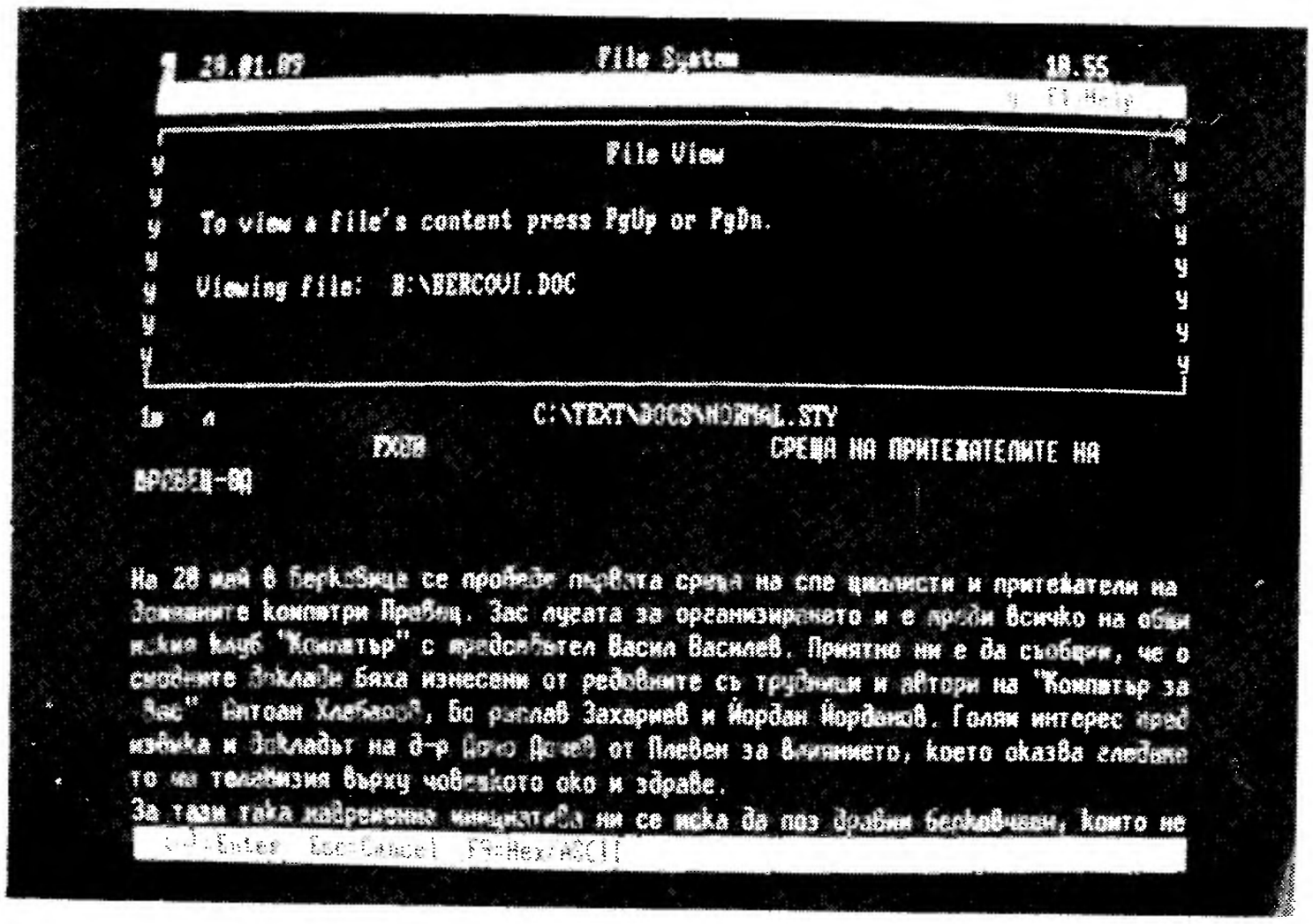
Файловете могат да се сортират по име, разширение, дата и големина (фиг. 7), както и да се разгледа тяхното съдържание — било в хексакод (фиг. 8), или аскикод (фиг. 9).

Освен графичната надстройка са направени и редица повече или помалко значителни промени в отделните команди на PC DOS 4.0. Достатъчно е да се погледне на фиг. 10[1], където е показано съдържанието на стартовите (системните) дискети ия двете версии 3.30 и 4.0. Големината на командния интерпретатор COMMAND. COM и двата системни файла IBMBIO. COM И IBMDOS. COM е увеличена общо с 28 Кбайта. Файловете на DOSSHELL заемат допълнително още около 81 Кбайта. PC DOS 4.0 (без DOSSHELL)

[1] *Илюстрацията е и намек за скорошната поява на новата версия на популярния вече Команден организатор, с която по-подробно ще ви запознаем в следващия брой.*



Фиг. 8



Фиг. 9

заема в оперативната памет 6 Кбайта, което е с 9 Кбайта повеч от DOS 3.30.

Усъвършенствани са и повечет от командите на ДОС, добавен са и нови. Някои от подобрения са дребни наглед, но много полез ни. Така например, ако командат за изтриване на файлове DEL [ete се зададе с параметър /p, компют ърът издава предупредително съоб щение, преди да се пристъпи къ изтриване. Ускорен е и достъпът д записаната на дисковете информа ция чрез рационално използван на буфери и „кеш-памет“.

PC DOS 4.0 е в състояние бе помощни програми да управлява и разширена оперативна памет (Expanded Memory) по стадарта Lotus-Intel-Microsoft.

Другото значително нововъве дение, което липсва и при първата версия на OS/2, е, че при PC DOS 4.0 е преодоляна досега съществу ващата пределната граница от 32 Мбайта за големината на дела за ДОС върху запаметяващото устройство. На какво се дължи това ограничение? Известно е, че след форматирането на диска по електромагнитен път върху него се очертават множество концентрични пътечки, всяка от които е разделена на сектори. Докато броят ия пътечките е определен хардуерно, то броят и големината на секторите може да се променят по програмен път по време на физическото форматиране на диска. PC DOS стандартно поддържа големина ия секторите 512 байта, макар че BIOS позволява и други големини — 128, 256 и 1024 байта. За номериране на секторите ДОС използва 16-битови числа, което определя и възможния брой на секторите — 65535. При стандартната големина от 512 байта пределният капацитет на дела за ДОС е 32 Мбайта.

Тази граница лесно може да бъде преодоляна, като се увелнчи големината на секторите, но веднага възникват проблеми със съвместимостта. Към уинчестърите на Seagate например има дискета за тяхното форматиране Disk Manager, с която дискът се форматира на два дяла: единият — 800 Кбайта за ДОС, а другият — с по-големи сектори, и така се преодолява ограничението, наложено от ДОС. Това също е компромисно

B:\

Име		Размер	Дата	Време
4201	cpi	17089	6.07.87	19:34
5202	cpi	459	6.07.87	19:34
ibmbio	com	22169	6.07.87	19:31
ibmdos	com	30159	6.07.87	19:31
ansi	sys	1678	6.07.87	19:31
autoexec	bat	162	3.04.88	12:40
bds-soft	com	1320	13.11.85	11:29
click	com	896	18.01.87	13:25
clock	com	1536	6.07.87	19:35
command	com	25979	30.09.87	21:10
config	sys	63	6.08.87	12:14
config1	sys	41	6.07.87	19:31
country	sys	11285	6.07.87	19:31
display	sys	11365	6.07.87	19:31
dosedit	com	1706	6.07.87	19:35
driver	sys	1204	6.07.87	19:31
ega	cpi	49065	6.07.87	19:34
fastopen	exe	3935	6.07.87	19:31

78.307 байта в 3 избрани файла

A:\ 10:0

Име		Размер	Дата	Време
4201	cpi	6404	17.06.88	12:00
4208	cpi	641	17.06.88	12:00
5202	cpi	402	17.06.88	12:00
Dos031	400	0	17.06.88	12:00
ibmbio	com	32816	3.08.88	12:00
ibmdos	com	36000	3.08.88	12:00
ansi	sys	9148	17.06.88	12:00
append	exe	11186	3.08.88	12:00
autoexec	bat	79	1.01.89	0:19
command	com	37637	17.06.88	12:00
config	sys	139	1.01.89	0:19
country	sys	12838	17.06.88	12:00
display	sys	15741	17.06.88	12:00
driver	sys	5274	17.06.88	12:00
graftabl	com	10271	17.06.88	12:00
graphics	com	16733	17.06.88	12:00
graphics	pro	9413	17.06.88	12:00
keyb	com	14759	17.06.88	12:00

106.453 байта в 3 избрани файла

A:

PC DOS 3.30 PC DOS 4.0

Фиг. 10

решение, защото се налага използването на допълнителен драйвер и т. н.

Увеличаването на броя на секторите изисква по-голям числен обхват за адресиране на секторите, което пък е свързано със сериозни промени във вътрешните структури от данни в ДОС, имащи отношение към управлението на дисковите устройства.

Именно така е постъпено в PC DOS 4.0. За адресиране на секторите се използват 32-битови числа, което при запазване на стандартната дължина на секторите позволява теоретичната граница на капацитета на дисковото устройство да се измести до 2048 Гбайта. Казахме „теоретична“, защото има и други ограничения, наложени от вътрешната структура на ДОС, в резултат на което максималната големина на дела за ДОС е 2 Гбайта. Макар и уголемена, ограничения налага и таблицата за разположението на файловете (FAT), в резултат на което големината на дела при сектори по 512 Кбайта и големина на кластърите по 4 сектора (за 20-Мбайтов уинчестър) вече възлиза на максималното 128 Мбайта. За големина на дела 2 Гбайта големината на кластърите трябва да бъде по 64 сектора, което пък е свързано с доста разхитително използване на дисковото пространство.

Възможността за директно управление на многократно по-голям обем външна памет обаче не е минала съвсем безнаказано. Възникват проблеми при използването на някои съществуващи програмни продукти при работата им под управление на PC DOS 4.0. Появата на такива трудности естествено не е новост и именно „съвместимостта“ често пъти се явява и главната преграда пред развитието на системния софтуер и техническите нововъведения.

MS-PC DOS се е наложил като индустриален стандарт в света на персоналните компютри и тази съвместимост е задължителна. Според фирмата „Майкрософт“ всички „добре“ написани програми, които не използват специални програмистки трикове, би трябвало да работят нормално под управление на PC DOS 4.0. На практика обаче се оказва, че някои твърде полезни и затова популярни продукти като Norton Utilities и PC-Tools използват именно такива „хватки“ за достъп до информацията на диска при извършване на някои важни операции като реорганизация и оптимизиране на записите, тяхното редактиране, при възстановяване на изтрити файлове и т. н. А едва ли има уинчестър-диск, на който да липсват тези два продукта. Проблеми са възникнали и при използването на някои други продукти, предимно предназначени за обслужване на мрежи.

Трудности със съвместимостта възникват и от промяната във функциите на ДОС за директно четене и записване от и върху диска (прекъсвания 25H и 26H), както и от факта, че при форматиране или копиране с DISKCOPY PC DOS 4.0 поставя задължителен сериен номер, получен от моментната стойност на датата и часа. ДОС установява смяната на дискетата по този сериен номер, а това може да създаде проблеми при дискети, форматирани с по-раншни версии на PC DOS или с някои служебни програми.

PC DOS 4.0 е несъвместим и с версията 1.0 на OS/2.

За преодоляване на проблемите, предизвикани от въвеждането на новия ДОС, вече има версии на споменатите по-горе пакети с помощни програми[2], съобразени с промените в PC DOS 4.0, разработват се съответни версии и на други продукти. И все пак в някои чуждестранни публикации съветват да не се избързва с инста-

[2] Norton Utilities Adv. Edition *4.50 е вече позната у нас.*

лирането на новия ДОС, който вероятно не е изчистен напълно и от възможни грешки.

При тестването му той отказа да работи и с някои от разпространените драйвери за кирилизиране на клавиатурата!

Този съвет е още по-разумен за повечето от нас, които не разполагаме нито с твърд диск с подобаващ капацитет, нито с компютър или пък принтери (че даже и мишки), които да могат да ползват възможностите на новия ДОС.

Колкото до графичния интерфейс, едва ли бихте изпитали кой знае каква потребност от него, след като сте заредили в паметта Командния организатор (заема памет само 15 Кбайта), а Norton Utility е винаги подръка.

Със създаването на PC DOS 4.0 бе отговорено и на въпроса, не означава ли появата на OS/2 краят на досегашния световен стандарт, наложен от MS DOS. Този край очевидно далеч още не е настъпил и няма така скоро да дойде. Според възможностите на компютърната система и режима на нейното използване двете операционни системи биха могли още дълго да съжителстват и да се допълват.

Дотук най-общо ви представихме PC DOS 4.0. Така, както той се представя пред по-втренчения поглед на потребителя. Споделеният опит е придобит след неколкодневно общуване с PC DOS 4.0 и както често пъти се иалага — по метода на пробите и грешките. По-интересно ще бъде обаче как той ще се държи на наша, българска територия. Да сподели своите впечатления и опит в „опитомяването“ и „национализирането“ на новия ДОС ще дадем думата в един от следващите броеве на нашия постоянен сътрудник и консултант **Орлин Вълчев.**

□

ОБРАЗОВАНИЕ

Един от най-трудните и бавни процеси в управлението на училището е разпределението на часовете на учителите — да се състави така наречения списък „Образец 1“. Тази дейност отнема значителна част от времето на директора, на учителите и техническия персонал. Предварителните пресмятания се извършват през февруари — март, а окончателното предаване на „Образец 1“ за утвърждаване става през септември. Това е основният финансов документ и от прецизността му зависи до голяма степен цялостната организация на учебната и финансово-счетоводната дейност. Неудобното време на изготвяне на окончателния вариант (септември) и отговорността на работата внасят допълнително напрежение.

„ОБРАЗЕЦ 1“ ЗА УЧИЛИЩЕТО

Опитите да се използва готова електронна таблица за решаване на проблема се оказаха безуспешни. Това ни накара да направим по-прецизно изследване, за да установим причините. Оказа се, че обемът на данните, с които трябва да се оперира, е твърде голям, за да може да се обхване с помощта на готова електронна таблица за осембитов компютър. Освен това у нас няма специализиран програмен продукт за решаването на този проблем.

Необходимостта да се рационализира тази дейност доведе до създаването на оригинален програмен продукт, ориентиран към фамилията Правец-82. В момента това са най-разпространените компютри в училищата. От подобни съображения за програмен език бе избран Разширен Бейсик.

През 1986/1987 учебна година бе разработен първи вариант на програмата и с нейна помощ бе проверен ръчно изготвеният „Образец 1“. След отчитането на редица технически и психически фактори бе изготвен втори вариант на програмата. През учебната 1987/1988 година тя бе използвана при изготвянето на образеца, като отново бе извършено и ръчното му обработване. Успоредно с това бе уточнена и технологията на компютърната обработка.

В сегашния момент са разработени напълно работоспособна програма и цялостна технология за обработка чрез „Образец 1“. Етапите на работата са следните:

● **Въз основа на учебните планове и програми се изготвят изходните таблици, като се отчитат настъпилите промени. Предвидена е възможност да се отрази и броят на групите, на които се дели дадена паралелка по съответния предмет. С друго подреждане, но също комбинирани са и отпечатаните таблици.**

● **Въвеждат се изходните данни на учителите: норматив, заплащане за един лекторски час и горна граница на лекторските часове. На този етап е добре да се направи копие на дискетите.**

● **Таблиците и началните записи на учителите се отпечатват и се предоставят на директора и**

...-директорите за проверка. След извършване на поправките информацията се отпечатва отново и се предоставя на предметните обединения.

● Предложенията на обединенията се въвеждат в компютъра. Тези корекции могат да се извършват и от техническо лице.

● Проверката и поправките могат да се правят направо на компютъра. Това става бързо и не е затруднително.

● Прави се рекапитулация на разпределените часове. Таблиците се преглеждат в автоматичен режим и се отпечатва списък на предметите и останалите неразпределени часове по всеки от тях, общата сума на разпределените часове, общата сума на неразпределените часове, сумата на лекторското заплащане в левове.

● Таблиците, в които са останали часове, могат да се разгледат на екрана на видеомонитора или да се отпечатат. Входно-изходните модули на програмата са конструирани така, че в най-голяма степен да облекчават потребителите. Програмата работи в диалогов режим. Проверяват се типът на всички входни данни, дължината на знаковите низове, долната и горната граница на въвежданите цифрови данни. Екраните са защитени от неправилно манипулиране.

Изключена е всяка възможност за неволно изтриване на данни. При аварийни ситуации (спиране на електрозахранването или повреда на дискетата) данните могат да бъдат възстановени сравнително лесно. Предвидена е възможност при брой на паралелките, помалък от максимално възможния, размерът на таблиците да се намали, което ускорява работата както при обработката на данните, така и при отпечатването им.

При форматиране на дискетите се задава броят на класовете. Посочват се номерацията им с букви и римски цифри, учебната година, името на училището, паролите за достъп. След това таблиците се форматират в автоматичен режим и са готови за работа.

Времето за изготвянето на „Образец 1“ през първата година е съизмеримо с времето за работа по традиционния начин. През следващите години то е вече около 10 пъти по-малко. За броени часове таблиците се поправят и отпечатват. А новите записи на учителите се въвеждат за един нормално уплътнен работен ден. Остава достатъчно време за събеседване с учителите. На всеки преподавател се предоставя листинг, върху който са отразени: нормативът му, сумата на нередуцираните часове, заплащането на един лекторски час, сумата на лекторските часове, сумата на лекторските възнаграждения в левове и по кой предмет, на кой курс и на кои групи ще преподава. По този начин се гарантират максимална прецизност и прегледност на крайните резултати. Справките могат да се извършат направо на компютъра или чрез направените листинги.

За работа с програмата не се изисква никаква специална подготовка. След включването на компютъра тя се стартира автоматично, като на всеки екран се изписват съответните подсказки.

Инж. ДИМИТЪР ШИШМАНОВ

Бележка на редакцията:

Читателите, които искат да получат по-подробна информация за „Образец 1“, могат да се свържат с автора на адрес:

гр. Бургас
к-с „Славейков“, бл. 16, вх. 7, ап. 9
Димитър Шишманов

□

КНИГОПИС

КНИГИ ОТ БЪЛГАРСКИ АВТОРИ

* Павлов, Д., Д. Чуровски, З. Върбанов. Педагогически измерения на компютризацията. С., Народна просвета, 1988, 191 с. (Поредица „Компютърът и образованието“, кн. 1).

КНИГИ, ПРЕВЕДЕНИ НА БЪЛГАРСКИ

* Дийн, К., К. Унтлок. Наръчник по компютърно обучение. С., Наука и изкуство, 1988, 276 с.

КНИГИ ОТ СЪВЕТСКИ АВТОРИ

* Баранов, С. Н., Н. Р. Ноздрунов. Язык Форт и его реализация. Л., Машиностроение, ЛО, 1988, 158 с.
* Вигдорчик, Г. В., А. Ю. Воробьев, В. Д. Праченко. Основы программирования на ассемблере для СМ ЭБМ. Второ изд. М., Финансы и статистика, 1987, 240 с.
* Костюк, В. И., А. И. Дешко, Б. В. Игнатенко. Проектирование информационных моделей в гибких системах. Киев, Изд. КГУ издат. объед. Вища школа, 1987, 176 с.
* Молчанов, И. Н. Машинные методы решения прикладных задач. Алгебра, приближение функций. Киев, Наукова думка, 1987, 288 с.
* Шабанов-Кушнаренко, Ю. П. Теория интеллекта. Проблемы и перспективы. Харьков, Изд. ХГУ издат. объед. Вища школа, 1987, 160 с.

КНИГИ, ПРЕВЕДЕНИ НА РУСКИ

* Браун, П. Введение в операционную систему UNIX. М., Мир, 1987, 288 с. (Серия „Математическое обеспечение ЭВМ“).
* Дейт, К. Руководство по реляционной СУБД ДВ2. М., Финансы и статистика, 1988, 320 с.
* Коффрон, Дж., В. Лонг. Расширение микропроцессорных систем. М., Машиностроение, 1987, 320 с.
* Морсита, И. Аппаратные средства микроЭВМ. М., Мир, 1988, 280 с.
* Рафикузаман, М. Микропроцессоры и машинное проектирование микропроцессорных систем. М., Мир, 1988, 312 с.

□

-5 03-04 89

ЕЛЕКТРОННА ПОЩА

ВЕСЕЛИН КОСТОВ
Инж. ТЕОДОР ПРЕВАЛСКИ
Н. с. инж. ВЕСЕЛИН БОНЧЕВ

Въпреки названието си „персонални“ компютрите у нас твърде често се използват от повече хора. Тъй като нито хардуерът (имаме предвид Правец-16 и микропроцесора 8086/8088), нито софтуерът (ДОС-16) улесняват работата в т. нар. многопотребителски режим, напълно естествено е, че на практика „миогото потребители“ ие работят едновременно, а един след друг.

Това ограничение изисква присъствието на едно много полезно средство за комуникация, което притежават почти всички многопотребителски системи. Това средство се нарича *електронна поща* и позволява на отделните потребители да обменят съобщения чрез използването на компютъра като междинно средство, т. е. като пощенска кутия.

В нашия случай електронната поща е реализирана почти изключително само със средствата, които предлага програмният език на .BAT-файловете. Единствената допълнителна програма, чието използване е наложително, е програмата WHAT. Тя позволява на потребителя да въведе знак или цял низ. Въведеният низ се запомня като стойност на променливата WHAT от обкръжението (environment). Програмата WHAT се разпространява във вид на изходен текст с дискетите на пакета MASM на фирмата „Майкрософт“. За тези от вас, които не могат да я намерят, в редакцията ще бъде предоставена на разположение дискета, която ще съдържа всички файлове, за които става дума в тази статия.

Преди да преминем към обяснението, как работи предлаганата електронна поща — една последна забележка. При създаването ѝ сме се стремили да моделираме действието на командата mail в ОС UNIX — една добре известна, широко разпространена и, трябва да отбележим, достойна за подражание операционна система. Това моделиране обяснява английския текст на използваните съобщения.

Електронната ни поща се състои от два основни .BAT-файла — LOGIN.BAT (листиг 1) и MAIL.BAT (листинг 2). Ако ги въвеждате от посочените листинги, трябва да пропуснете номерацията на редовете — тук тя е използвана, за да улесни обяснението им. За въвеждането на файловете можете да използвате който и да е редактор, способен да създава обикновени текстови файлове — PE2, E3, ED или дори EDLIN.

Тук ияма да се впускаме в подробно обяснение на езика на .BAT-файловете — за справка вижте КВ.09.87, а за детайлно описание на такива хватки, като

```
if x%1 == x ...
```

можете да ползвате отличната статия на Орлин Вълчев (КВ.01—02.89).

Файлът LOGIN.BAT трябва да се изпълнява винаги, когато пред компютъра седне нов потребител. Целта на файла е именно да регистрира този нов потребител и да провери дали на негово име има оставена поща. Съветваме ви да включите обръщение към LOGIN.BAT като последен ред от файла AUTOEXEC.BAT. По този начин, след всяко рестартиране, системата ще се осведомява за името на работещия с нея потребител.

Листинг 1

LOGIN.BAT

```
1   echo off
2   set mail=c:\usr\spool\mail
3   if x%1 == x goto noname
4   set what=%1
5   goto cont
6   :noname
7   what s "login: "
8   :cont
9   set user=%what%
10  set what=
11  cd\usr\%user%
12  if not exist %mail%\%user% goto nomail
13  set oldp=%prompt%
14  set prompt=You have mail$ %prompt%
15  :nomail
```

MAIL.BAT

```
 1  echo off
 2  if x%1 == x goto getmail
 3  if exist %mail%\%1 copy %mail%\%1+nul %mail%\%1 /b > nul
 4  echo Message from %user% to %1 >> %mail%\%1
 5  echo Enter message to %1:  Z RETURN  to quit:
 6  copy %mail%\%1+con %mail%\%1 > nul
 7  echo Done.
 8  goto quit
 9  :getmail
10  if not exist %mail%\%user% goto nomail
11  more < %mail%\%user%
12  what ce "Remove mail ? (Y/N) " yn
13  echo %what%
14  if %what% == N goto nodel
15  del %mail%\%user%
16  set prompt=%oldp%
17  set oldp=
18  :nodel
19  set what=
20  goto quit
21  :nomail
22  echo You have no mail
23  echo usage: %0 [user]
24  :quit
```

Листинг 2

Ред 1 спира извеждането на екраиа на изпълняваните от файла команди.

Ред 2 записва в променливата MAIL от обкръжението директорията, в която ще се записват съобщенията, т. е. пощенската кутия. В нашия пример името е същото, както и в системата UNIX [\usr\spool \ mail), но при вас, разбира се, то може да е съвсем друго.

На ред 3 се проверява дали иа програмата, намираща се във файла LOGIN.BAT, е подаден аргумент. Ако това е така, стойността му се записва в променливата WHAT от обкръжението (редове 4 и 5). В противен случай (ред 6) се използва програмата WHAT, която извежда подсещащото съобщение „login:“ (ред 7) и очаква въвеждането на низ от потребителя (опция „s“). Въведеният низ се записва в променливата WHAT. Този алгоритъм позволява името на потребителя да се посочи директно като аргумент на програмата LOGIN — без да се задават въпроси. Специално внимание трябва да се обърне на факта, че при въвеждането на името на потребителя се прави разлика между главни и редовни букви. Така например Otto и otto са две *различни* имена на потребители.

На ред 9 съдържанието на променливата WHAT се прехвърля в променливата USER на обкръжението, а самата WHAT (която се използва само като работна) се унищожава (ред 10).

Командата от ред 11 установява текуща директория за съответния потребител. Тук се предполага, че всеки регистриран потребител разполага със собствена директория, която е поддиректория на \ usr. Разбира се, ако при вас това не е така или ако поради някакви други причини не желаете да се извърши смяна на текущата директория, тази команда трябва да бъде пропусната.

На ред 12 се проверява дали в директорията, служеща за пощенска кутия (стойността на променливата MAIL), има „писмо“ — файл с името на текущия потребител. В този случай системният промпт се променя, като отпред му се вмъква съобщението „You have mail<нов_ред>“ („Имате поща“).

Това се извършва на ред 14. С цел по-късно да може да бъде възстановен, оригиналният промпт се записва в променливата OLDP от обкръжението (ред 13). Тук трябва да се отбележи, че командите от редове 13 и 14 ще работят ко-

ректно само при условие че в стойността на текущия промпт не се съдържа знакът „=“. Ако това условие не е изпълнено, командите на тези редове ще предизвикат съобщението „Syntax error“. Това няма да попречи на по-нататъшната работа на системата, но промптът няма да бъде установен правилно и потребителят ияма да бъде предизвестен за наличието на поща.

Програмата, намираща се във файла MAIL.BAT, служи за две цели: изпращане на съобщение до друг потребител и прочитане на съобщението, изпратено от текущия потребител. В първия случай като аргумент се подава името на потребителя, до който трябва да се изпрати съобщението:

```
mail < user_name >
```

а във втория случай програмата се извиква без аргументи:

```
mail.
```

Първият ред от файла изключва извеждането на комаидите на екрана. На ред 2 се проверява дали програмата е била извикана с аргумент. Ако такъв отсъства, управлението се предава на ред 9. В противен случай се продължава с изпълнението на ред 3.

Последният изисква малко повнимателно разглеждане. Задачата му е да премахне евентуално присъстващия знак^Z в края на файла, съдържащ предишни съобщения. За целта файлът се копира в себе си, като се чете (по подразбиране) като текстов (^Z не се прочита), а се записва (опция „/b“) като двоичен (^Z не се добавя).

Наставката +nul служи, за да бъде заблудена командата copy, която отказва да копира файл в себе си. Самите съобщения на copy се пренасочват в nul, за да не задръстват излишно екрана.

Ред 4 добавя в пощенската кутия ред, установява от кого и до кого е съобщението, а ред 5 извежда на екрана подсещащо съобщение за въвеждаие на „писмото“. То се въвежда от клавиатурата (con) и се добавя към старото съдържание на пощенската кутия (ред 6). След завършване на въвеждането на екраиа се появява съобщеиието „Done.“ (ред 7) и програмата прекратява работата си (редове 8 и 24). Ред 7 може да бъде пропуснат.

Ако програмата е извнкана без аргументи, предполага се, че потребителят иска да разгледа изпратените до него съобщения. Проверява се дали има такива (ред 10) и ако това не е изпълнено, извежда се съобщение за грешка (редове 22 и 23). Ред 23 може да изглежда и така:

```
echo usage: mail [user]
```

но ако решите да преименувате файла MAIL.BAT (странно хрумване!), този ред няма да отговаря на действителността. А така, както е показано на листинга, аргументът %0 винаги ще се замества с текущото име на изпълнявания файл.

Ако се установи, че пощенската кутия ие е празна, съдържанието ѝ се извежда иа екрана през филтъра MORE (ред 11), който позволява разглеждането да става на страници. MORE е още една програма, чието наличие е необходимо за работата на системата, но тя може да се смята за стандартиа, тъй като влиза в набора външни команди, които се разпространяват със системните дискети на ДОС-16.

Разбира се, на това място можете да използвате коя да е друга програма (иапример SHOW, която се разпространява заедно с WHAT) или дори текстов редактор!

След като разглеждането завърши, на потребителя се задава въпрос, дали да се премахне съдържанието на пощенската кутия (редове 12 и 13). Отново се използва програмата WHAT. В случая опциите „се“ означават, че се очаква въвеждането иа знак („y“ или „n“), при това — без ехо на екрана. Разбира се, редове 12 и 13 могат да се заменят с един-единствен:

```
what c „Remove mail ? (Y/N)“ yn
```

За съжаление програмата WHAT не преминава на нов ред след въвеждането на поискания знак. Така, след завършване на програмата MAIL, системният промпт се появява на същия ред, непосредствено след съобщението „Remove mail ? (Y/N) “. Това не е особено приятно от естетична гледна точка. От друга страна, преминаването на нов ред не може да се извърши с помощта на празен оператор echo, тъй като, когато е използван без аргументи, този оператор извежда текущото състояние на режима (в случая — ECHO OFF).

Ако потребителят отговори с „N“ на зададения въпрос (или с „n“; редовните букви се превръщат в главни от програмата WHAT), управлението (ред 14) се предава на ред 18. Там променливата WHAT се изключва от обкръжението (ред 19) и програмата завършва работата си (редове 20 и 24).

Обратно, ако отговорът е „Y“, в допълнение се изтрива файлът — пощенска кутня (ред 15). Възстановява се старият системен промпт (ред 16), записан в променлнвата OLDP, а самата променлива OLDP се изключва от обкръжението (ред 17).

В заключение ще направим две забележки. Първо, за да работи описаната електронна поща, необходимо е в обкръжението да има достатъчно място за добавянето иа променливите WHAT, MAIL, USER и OLDP. Второ, ако работите с PC-DOS, версия 3.30 или по-голяма (или SPS/ДОС-16; виж КВ.09—10.88), заменете операторите

```
echo off
@echo off
```

Това ще изключи и извеждането на първия ред от .BAT-файловете, което ще направи работата ви със системата още по-приятна.

ОБЯВА

Притежавам микрокомпютър Комодор–16, но за съжаление ие мога да открия никакво програмно осигуряване за иего. Желая да кореспоидирам с всеки, който има възможност да ми помогне.

4700 СМОЛЯН
обл. Пловдивска
кв. Райков
ул. Мирчо Войвода 2
Савелин Вел
тел. 2-44-91, код: 030

Прозорци в МИКРОФАЙЛ/16

16

Н. с. МАЯ ИВАНОВА
Н. с. ДИНКО ХРИСТОВ

Широко разпространение в света на 16-битовите микрокомпютри получи специализираният програмен продукт на СП „Микросистеми" — Микрофайл/16. При всичките си предимства обаче той не позволява работа с прозорци в потребителските програми. Донякъде този пропуск е компенсиран от програмния продукт ГЕПАРД на същата фирма, но неговите възможности за създаване на прозорци са ограничени само в сферата на менютата. Ето защо на читателите на „Компютър за вас" предлагаме прост и ефективен модул за организиране на прозорци със средствата на Микрофайл/16.

Възможностите на модула са:

- **извежда информационен прозорец;**
- **отпечатва съдържанието на избрано поле от файл от БД;**
- **извежда прозорец за избор;**
- **стойностите от полетата могат да се избират чрез движението на светлинен показалец.**

Тези предимства могат да се демонстрират чрез главната програма WIN.PRG, която се вгражда в информационните потребителски системи. Главната програма рисува екран (фиг. 1), в който се отварят прозорците.

```
ПРОГРАМА, ДЕМОНСТРИРАЩА РАБОТАТА С ПРОЗОРЦИ
В СРЕДАТА НА МИКРОФАЙЛ/16

име:      Стоян          фамилия: Стояновски

ЕГН:      5506094554

адрес:    вул."Ленин"132

телефон:  656556
                    ПОМОЩ

F1 = помощ  PgUp = пр.стр.
ESC = отказ PgDn = сл.стр.
          = избор

Стояновски
Илиев
Иванов
Драганов
Георгиев
Иванов
Симеонов
Петрова
Делчева
Палийски
```

Фиг. 1

Самият прозорец се организира от програмата WINDOW.PRG. Параметризирането й я прави независима от:

— конкретиата БД;
— дължината на полето от БД;
— координатите на прозореца;
— големината на прозореца.

Най-съществената част от програмата е проверката за натиснати управляващи клавиши чрез функцията INKEY(). Тази проверка продължава в цикъл, докато не бъде натиснат един от клавишите:

— RETURN (INKEY()=13) — направен е желаният избор от полето;

— ESC (INKEY()=27) — потребителят се е отказал от помощта на прозореца. При натиснат клавиш | (INKEY()=24) или | (INKEY()=5) светлинният показалец се движи в рамките на страница от прозореца (осъществява се SCROLL). При натиснат клавиш PgUp (INKEY()=18) или PgDn (INKEY()=3) се разлистват отделните страници в прозореца. Естествено при всеки отделен случай се проверява дали са достигнати границите на файла.

Тази програма може да бъде приложена и за филтрирано търсене на информация. По зададени част от характеристиките на поле е възможно в прозореца да се извикат всички обекти, които ги притежават. Така потребителят може да улесни избора си, когато не разполага с пълна информация за обекта.

```
* Програма.: WIN.PRG
* Автори...: Мая Иванова, Динко Христов
* Запазени.: begin,row,lenth,recno,name

*****************************************************************
* Тази програма служи само за демонстриране на работата с прозорци *
*              в средата на МИКРОФАЙЛ 16                        *
*****************************************************************

* Затваряне на всички отворени файлове
  CLEAR ALL
* Установяване на работната среда
  SET ESCAPE OFF
  SET TALK OFF
  PUBLIC recno ,begin,row,lenth,NAME
* Отваряне на потребителски файл Model.dbf
  USE model
  GO BOTT
  recno = RECNO()
  GO TOP
* Задаване на координатите на прозореца
  begin = 11
  row = 50
  lenth = 10
  name = SPACE(10)
  print = "____________________ [illegible] ____________________"
* Проверка за край на работа на програмата
  DO WHILE READKEY() <> 12 .AND. READKEY() <> 268
      CLEAR
    * Инициализация на променливите от паметта
      STORE SPACE(15) TO ime_m,fam_m,adr_m
      STORE 0 TO egn_m,tel_m
    * Рисуване на потребителски екран
      SET COLOR TO w/b
```

```
   @ 1,10 clear TO 20,70
   @ 1,10 to 20,70
   @ 4,20 SAY [ПРОГРАМА, ДЕМОНСТРИРАЩА РАБОТАТА С ПРОЗОРЦИ]
   @ 5,29 SAY "В СРЕДАТА НА МИКРОФАЙЛ/16"
   @ 8,15 SAY "име:"
   @ 8,43 SAY "фамилия:"
   @ 10,15 SAY "ЕГН:"
   @ 12,15 SAY "адрес:"
   @ 14,15 SAY "телефон:"
   @ 15,11 SAY print
   @ 17,23 SAY "F1 = помощ   PgUp = пр.стр."
   @ 18,23 SAY "ESC = отказ  PgDn = сл.стр."
   @ 19,23 SAY "        ↵ = избор        "
   @  8,24 GET ime_m PICT 'XXXXXXXXXXXXX'
   @  8,52 GET fam_m PICT 'XXXXXXXXXXXXXXX'
   READ
* Проверка дали е натиснат клавиш F1
   IF READKEY() = 36 .OR. READKEY() = 292
         * Наименование на полето , което ще се разпечатва в прозореца
           name = 'fam'
         * Обръщение към програмата, реализираща
прозореца                  DO window
         * При отказване от прозореца, връщане в първоначалния екран
           IF name = ' '
                    SET COLOR TO
                    LOOP
           ENDIF
         * Отпечатване на съдържанието на избрания от
прозореца запис                  SET COLOR TO  w/b
          @ begin,row CLEAR TO 20,70
          @  8,24 SAY ime
          @  8,52 SAY fam
          @ 10,24 SAY egn
          @ 1,10 to 20,70
          @ 15,11 SAY print
       ELSE
          @ 10,24 GET egn_m PICT '9999999999'
          READ
       ENDIF
       @ 12,24 GET adr_m PICT 'XXXXXXXXXXXXXXXXXXXX'
       READ
       IF READKEY() = 36 .OR. READKEY() = 292
          name = 'adr'
          DO window
          IF name = ' '
                    SET COLOR TO
                    LOOP
          ENDIF
          SET COLOR TO  w/b
          @ begin,row CLEAR TO 20,70
          @ 12,24 SAY adr
          @ 1,10 to 20,70
          @ 15,11 SAY print
       ENDIF
       @ 14,24 GET tel_m PICT '999999'
       READ
       SET COLOR TO
   ENDDO
   SET COLOR TO
   RETURN
* EOF: Win.prg



* Програма.: WINDOW.PRG
* Автори...: Мая Иванова, Динко Христов
* Дата.....: 16/06/88
* Забележки: Проектиране на прозорец и избор на запис
* Запазени.: begin,row,lenth,recno,name

* begin,row -- начални координати на прозореца
* lenth -- дължина на прозореца
* recno -- брой записи във файла
* name -- име на полето
* k -- номер на страница в прозореца
* p -- стойност на INKEY()
k = 1
p = 0

GO TOP

* Цикълът се изпълнява, докато не е натиснат клавиш RETURN
DO WHILE p <> 13
   * Рисуване на прозореца
   i = begin
   SET COLOR TO W/G
   @  i,row CLEAR TO i+lenth+1,row+3+len(&name)
   @  i,row  TO i+lenth+1,row+3+len(&name) DOUBLE
* Отпечатва съдържанието на посоченото поле в "прозорец"(K-та страни
   DO WHILE RECNO() < (k*lenth+1) .AND. .NOT. EOF()
          @  i+1,row+2 SAY  &name
          i = i + 1
          SKIP
   ENDDO
   GO ((k - 1)*lenth + 1)
   i = begin
   SET COLOR TO N/W
   @ i+1,row+2 SAY  &name
   p = 0
 * Цикълът се изпълнява, докато не е натиснат клавиш RETURN
   DO WHILE p <> 13
          p = INKEY()

          DO CASE
* Тест за натиснатата стрелка нагоре
          CASE p = 5 & .NOT. BOF()
                SET COLOR TO W/G
                @ i+1,row+2 SAY  &name
                SKIP - 1
                IF BOF() .OR. RECNO() = (k-1)*lenth
                      IF k*lenth < recno
                            GO k*lenth
                            i = i + lenth
                      ELSE
                            GO BOTTOM
                            i = i + recno - (k - 1)*lenth
                      ENDIF
                ENDIF
                i = i - 1
                SET COLOR TO N/W
                @ i+1,row+2 SAY  &name
* Тест за натиснатата стрелка надолу
          CASE p = 24 & .NOT. EOF()
                SET COLOR TO W/G
                @ i+1,row+2 SAY  &name
                SKIP + 1
                IF EOF() .OR. RECNO() > k*lenth
                      GO ((k-1)*lenth+1)
                      i = begin - 1
                ENDIF
                i = i + 1
                SET COLOR TO N/W
                @ i+1,row+2 SAY  &name
* Тест за натиснат клавиш PgDn
          CASE p = 3 .AND. .NOT.EOF()
                SKIP + 1
                IF EOF()
                      k = 0
                ENDIF
                GO (k*lenth + 1)
                k = k + 1
                EXIT
* Тест за натиснат клавиш PgUp
          CASE p = 18 .AND. .NOT.BOF()
                SKIP - 1
                IF BOF() .OR. k=1
                      GO BOTTOM
                      k = INT(recno/lenth) + 1
                      EXIT
                ENDIF
                GO ((k-2)*lenth+1)
                k = k - 1
                EXIT
          * Тест за натиснат клавиш ESC
             CASE p = 27
                      name = ' '
                      RETURN
          * Тест за натиснат друг клавиш
             OTHERWISE
                      LOOP
             ENDCASE
       ENDDO
ENDDO
SET COLOR TO
@  begin,row CLEAR TO begin+lenth+1,row+3+len(&name)
RETURN
* EOF: WINDOW.PRG
```

ТОПЪЛ И СТУДЕН РЕСТАРТ

Н. с. магистър ***ВЛАДИМИР МАКАРИЕВ***

В Правец-16 е предвидена възможност за повторно стартиране на операционната система чрез едновременно натискане на клавишите Ctrl, Alt и Del. След разпознаване на тази комбинация базовата система за вход и изход (BIOS) записва на адрес 0040:0072 шестнайсетичното число 1234 и предава управлението на адрес FFFF:0000. От този адрес започват нейните диагностични подпрограми. Когато съдържанието на адрес 0040:0072 е нула (колкото е след включване на захранването), се изпълнява вътрешният тест на компютъра (имитират се включване и изключване на компютъра).

Понякога се налага повторно стартиране на операционната система от потребителска програма. За тази цел предлагам две 16-байтови команди WARM и COLD, съответно за топъл и студен рестарт от ниво пакетен файл. За да създадете командите, стартирайте Бейсик интерпретатора с команда BASICA, въведете и изпълнете програмата, показана на фиг. 1.

```
100 REM ****************************************
110 REM *   Топъл и студен рестарт              *
120 REM *      магистър Макариев               *
130 REM *      (с) София, 1988 г.              *
140 REM ****************************************
150 READ NM$,L
160 GOSUB 210
170 READ NM$,L
180 GOSUB 210
190 SYSTEM
200 END
210 PRINT"Създаване на команда ";NM$
220 OPEN NM$ AS #1 LEN=1
230 FIELD #1,1 AS D$
240 FOR I = 1 TO L
250   READ N
260   LSET D$ = CHR$(N)
270   PUT #1
280 NEXT I
290 CLOSE
300 PRINT"КОМАНДА ";NM$;" Е СЪЗДАДЕНА !"
310 RETURN
320 DATA WARM.COM,16
330 DATA 184, 64,  0,142,216,199,  6,114
340 DATA   0, 52, 18,234,  0,  0,255,255
350 DATA COLD.COM,16
360 DATA 184, 64,  0,142,216,199,  6,114
370 DATA   0,  0,  0,234,  0,  0,255,255
```

Фиг. 1
Програма за създаване на команди WARM и COLD

```
100 REM ****************************************
110 REM * Програма WARM.BAS (Топъл рестарт)    *
120 REM *      магистър Макариев               *
130 REM *      (с) София, 1988 г.              *
140 REM ****************************************
150 DEF SEG = &H40
160 POKE &H72,&H34
170 POKE &H73,&H12
180 DEF SEG = &HFFFF
190 CALL A
200 END
```

Фиг. 2 Примерна програма за топъл рестарт

```
100 REM ****************************************
110 REM * Програма COLD.BAS (Студен рестарт)    *
120 REM *      магистър Макариев               *
130 REM *      (с) София, 1988 г.              *
140 REM ****************************************
150 DEF SEG = &H40
160 POKE &H72,0
170 DEF SEG = &HFFFF
180 CALL A
190 END
```

Фиг. 3 Примерна програма за студен рестарт

Как изглеждат примерни програми за решаване на този проблем от средата на Бейсик, е показано на фиг. 2. и фиг. 3. И в двете програми след записването на адрес 0040:0072 на посочените по-горе стойности управлението се предава на адрес FFFF:0000. Това е осъществено с оператор CALL A. В случая променливата A по подразбиране има стойност 0. За дефиниране на текущия сегмент от оперативната памет е използван оператор DEF SEG.

ПОЩА

РАЗПОЗНАВАНЕ НА МОДЕЛА

За пръв път пиша до редакция и не зная как би трябвало да оформя писмото си, но се надявам, че ще извлечете от него това, което би представлявало интерес, и ще го преработите за публикация. Не съм „спец" в областта на компютрите, но все пак мисля, че „откритието" ми ще заинтересува и други читатели.

Мониторната програма, записана в ромовете на Правец-82 и Правец-8М, се различава от мониторната програма на Apple и някои програмни продукти, предназначени за Apple, не работят с Правец-8М (XTYPER и други от APPLE MECHANICS; PIXIT в режим демонстрация; GRAPHEXT DOS ненужно сменя клавиатурата, а програмата за функционални клавиши, която модифицира ДОС от адрес $9B00, въобще не работи). Тук предлагам „лечение" за последните две програми, тъй като са по-разпространени.

Ето една HELLO-програма за GRAPHEXT DOS:

```
10  IF  PEEK (62923) <  > 165 THEN
     POKE 40273,234: POKE 40274,
    234: REM  АКО КОМПЮТЪРЪТ НЕ
    Е ПРАВЕЦ-82, КЛАВИАТУРАТА НЕ
     СЕ ОБРЪЩА
20  HOME : PRINT : PRINT : PRINT
    : PRINT : PRINT : PRINT : PRINT
    "ЖЕЛАЕТЕ ЛИ ОЗВУЧЕНА КЛАВИАТ
    УРА?" +  CHR$ (8);: GET A$: IF
    A$ = "N" OR A$ = "Н" THEN  POKE
    46826,234: POKE 46827,234: POKE
    46828,234: REM   ПРИ ЖЕЛАНИЕ
     ПРЕМАХВА ОЗВУЧАВАНЕТО
30  HOME : NEW
```

И едно допълнение към програмата за функционални клавишн:

```
9AEF:   AD CB F5  LDA $F5CB     ;Компютърът е Правец-82?
9AF2:   C9 A5     CMP #$A5
9AF4:   F0 0A     BEQ $9B00     ;Ако да, без модификация
9AF6:   A9 DD     LDA #$DD      ;Ако не, модифициране
9AF8:   8D 78 9B  STA $9B78     ;1.Смяна на извеспяващия знак
9AFB:   A9 60     LDA #$60
9AFD:   8D 58 9B  STA $9B58     ;2.Клавиатурата не се обръща
9B00:   A9 9B     LDA #$9B      ;Начало на оригиналната програма
```



Оригиналното в двете програми е това, че „познават" дали компютърът изисква модифициране на продукта, или не, което би било полезно и в други случаи.

АЛЕКСАНДЪР НАЧЕВ
Враца

Бележка на научния консултант

Писмото на Александър Начев повдига един интересен въпрос. Това е въпросът за различаването по програмен път на различните версии на даден модел компютри. Обикновено това става чрез проверката на един или повече байта от постоянната памет на компютъра. Така например 8-битовите модели на фирмата „Apple" могат да се отличават помежду си най-много след две проверки. Характеристичните адреси и съдържанието им са показани в следната таблица:

Компютър	Адреси		
=====	$FBB3	$FBC0	$FB1E
Apple][	$38	$E0	
Apple][+	$EA	$E0	$AD
Apple //e	$06	$EA	
Apple //e	$06	$E0 $$	(модернизирана версия)

Аналогично Правец-82 и Правец-8М се различават по това, че в постоянната памет на Правец-82 на адрес $F5C3 е записан байтът $A5.

Н. с. ВЕСЕЛИН БОНЧЕВ

ОТГОВОРИ

Нашите читатели Чавдар Бончев и Христо Илиев от Кърджали задават следния въпрос (виж факсимилето на писмото):

Защо изразът

35.9 * 10 = INT (35.9 * 10)

има стойност 0, т. е. неистина? Още повече че командите PRINT 35.9 * 10 и PRINT INT (35.9 * 10) извеждат едно и също число — 359.

Работата се състои в това, че изразите от двете страни на знака за сравнение действително *не са* равни. В това можете да се убедите, като ги извадите един от друг: операторът

PRINT 35.9 * 10 — INT (35.9 * 10)

извежда

1.1920929E-07

Това означава, че за Правец-82 35.9 * 10 е равно на 359.00000011 920929. За съжаление операторът PRINT извежда само *осем* значещи цифри на резултата, което предизвиква заблуждаващия отговор 359.

Но все пак всички знаем, че 35.9 * 10 е равно на 359! Кое кара компютъра да пресмята погрешно?

Причината е, че винаги, когато работи с числа с плаваща точка, компютърът греши по малко поради закръгляването. И наистина — поради ограничения брой байтове, в които се представя числото с плаваща точка, числа с период (като например 1/3) по принцип не могат да се представят точно. При това не забравяйте, че компютърът работи в двоична бройна система. А в нея са периодични много дроби, които не са периодични в десетичната. Например 0.9 десетично е равно на 0.1 (1100) двоично. Затова и числото 35.9 не може да се представи точно.

Изложеното по-горе важи за *всички* компютри, а не само за Правец-82. Всички компютри грешат, независимо с каква бройна система работят; независимо по колко байта отделят за представянето на числата. Грешката възниква винаги при операции с плаваща точка (аритметични действия или сравнение) и тя се дължи на неизбежното закръгляване. Разбира се, *големината* на допусканата грешка зависи както от бройната система, така и от броя байтове, с които се представят числата. Но по-важното е, че грешка винаги има. За щастие в много случаи тя е пренебрежимо малка.

```
]A=INT(35.9*10)

]A1=35.9*10

]?A
359

]?A1
359

]ВНИМАНИЕ!!!

]?A=A1
0


]?359=359
1

]БЕЗ КОМЕНТАР...
```

МОЖЕ ЛИ КОМПЮТЪРЪТ ДА СМЯТА?

От казаното дотук може да се изведе едно важно правило. Никога *не разчитайте, че два израза са равни, ако стойностите ти са числа с плаваща точка.* А да не забравяме, че в реализацията на Бейсик за Правец-82 стойностите на *всички* изрази са числа с плаваща точка — дори и на тези, които съдържат само целочислени константи и целочислени променливи. Ако трябва да проверите два израза за равенство, *не използвайте*

IF A = B THEN ...

а

IF ABS (A — B) < EPSILON THEN ...

където EPSILON е константа, специфична за конкретния компютър. За Правец-82 тя е 1E-6.

Всъщност описаният проблем е много по-дълбок, отколкото може да изглежда на пръв поглед. На практика компютърната аритметика с плаваща точка силно се различава от аритметиката, която сме учили в училище. Нещо повече — тя има толкова странни свойства, че може би е твърде пресилено изобщо да бъде наричана аритметика. Съдете сами:

— Понятието *плюс безкрайност* и *минус безкрайност* не съществуват. Вместо тях съществуват *най-голямо* и *най-малко* число (за Правец-82 те са ±1.70141183E+38).

— Нулата не е единствена, а съществуват много т. нар. „машинни нули" (за Правец-82 всички числа в диапазона ±2.93873588E—39 са неотличими от нулата).

— Понятията *рационално число, трансцендентно число* и *периодична дроб* не съществуват — всички дроби са рационални.

— Числата върху числовата ос не са наредени плътно, както е при реалните числа, но и не отстоят на равни интервали, както е при целите числа. Наредбата е съвсем странна — малките (по абсолютна стойност) числа отстоят помежду си на малки интервали, а големите — на големи.

— Асоциативният и дистрибутивният закон не важат (на Правец-82 можете да пресметнете израза 1E38/20*10, но 1E38*10/20 предизвиква ?OVERFLOW ERROR).

— Като следствие от горното аритметичните действия се изпълняват неточно, със закръгляване.

Наистина в повечето случаи възникващите грешки са пренебрежимо малки. Но това не винаги е така! Понякога те се натрупват.

Нека да разгледаме следната задача. Вземете числото 1.0000001. Повдигнете го на квадрат. Резултата отново повдигнете на квадрат. Повтаряйте тази операция 27 пъти.

Ако имате калкулатор с клавиш за извършваие на операцията повдигане на квадрат, наберете числото 1.0000001 и натиснете този клавиш 27 пъти. Резултатът с 10 верни цифри е 674530.4707. Но вие със сигурност ще получите нещо друго.

Програмата на Бейсик, която решава горната задача, изглежда така:

```
10 X = 1.0000001: FOR I = 1 TO
27:X = X * X: NEXT : PRINT X
```

При изпълнението ѝ ще получите 22723.9709 — резултат, нямащ нищо общо с действителността.

Задачата е еквивалентна (математически еквивалентна, разбира се) на пресмятането на израза 1.0000001 ^ (2 ^ 27). Ако обаче го пресметнете именно по този начин, а не итеративно, ще получите друг резултат — 665348.189 (една вярна цифра).

На табл. 1 са показани резултатите от итеративното решаване на горната задача за различни компютри, различни езици и различна точност на представяне на числата. Вижда се, че резултатите силно се различават помежду си. И причина за всичко това е закръгляването.

Така че, когато ви зададат на пръв поглед тривиалния въпрос, може ли компютърът да смята, размислете, преди да отговорите.

Н. с. инж. ВЕСЕЛИН БОНЧЕВ

Таблица 1

Компютър	Език	Резултат (10 знака)
Casio fx-115M	(ел. калкулатор)	674294.1172
TI-59P	(ел. калкулатор)	674520.6053
Правец-82	Applesoft BASIC	27723.97090
Правец-82	UCSD Pascal	8850400.000
Правец-82	UCSD FORTRAN 77	8850400.000
Правец-82	Turbo Pascal v. 3.0	674423.3480
Правец-16	MS FORTRAN 77 (X — double precision)	8886102.045
Правец-16	MS FORTRAN 77 (X — real)	8850397.000
Правец-16	Turbo BASIC (X — real)	8850397.000
Правец-16	MS C (X — float)	8850396.938
Правец-16	GW BASIC (X — real)	8850273.000
Правец-16	Turbo Pascal v. 3.0	674423.3480
Правец-16	Turbo Pascal v. 4.0 (с опция {$N+})	674587.9853
Правец-16	Turbo C (X — double)	674530.4554
Правец-16	MS C (X — double)	674530.4755
Правец-16	Turbo BASIC (X#)	674530.4755
Правец-16	GW BASIC (X# — double)	674530.4706
Правец-16	Turbo Pascal v. 4.0 (X — extended)	674530.4707

Притежавам домашен компютър Правец-8Д. На платката до постоянната памет има допълнителен цокъл за включване на EPROM.

Възможно ли е включването на памет EPROM *2516J и ако е възможно, по какъв начин ще се осигури обслужването ѝ?*

Ивелин Василев, Шумен

На въпроса на Ивелин Василев отговаря нашият консултант инж. БОРИСЛАВ ЗАХАРИЕВ.

На платката на Правец-8Д са предвидени два цокъла за постоянна памет тип EPROM. Така компютърът може да се комплектова както с два чипа от типа 2764 (8 K × 8 bit), така и само с един чип 27128 (16 K × 8 bit). Замяната на едните чипове с други е съпроводена с установяване на съответни мостчета, намиращи се на платката. Първите компютри бяха предимно с два чипа, но по-късно (както е във вашия случай) заводът в Правец започна да поставя само една памет PROM тип 27128. Тъй като адресното пространство на постоянната памет е от адрес #C000 до #FFFF, поставянето на друг чип в свободния цокъл (без корекции на платката) би довело до конфликт с наличната памет — просто няма свободни адреси за нея!

ПОЩА

В много писма до редакцията читателите на КВ ни питат къде могат да намерят или ползват литература по различните езици за програмиране. Конкретният повод, който ни накара да се обърнем към нашия научен консултант Димитър П. Шишков, бе писмото на школника от НШЗО „Христо Ботев" в Плевен Найден Василев, който се интересува от програмните езици Си, Ада и Турбо Бейсик.

За съжаление литература за тях, а и за повечето други езици за програмиране се намира много трудно — и на български, и на руски език. Такива книги се разграбват веднага, въпреки че тиражите им никак не са малки. Но специално за читателите на „КОМПЮТЪР ЗА ВАС" нашият научен консултант подготви едни

Практически пълен списък на КНИГИ ПО СИ, АДА И ТУРБО БЕЙСИК (на български и руски език)

СИ

От български автори:

1. Си компилатор. СИ-16. С., ИТКР БАН и Комбинат по микропроцесорна техника, гр. Правец, 1986, 230 с. Разпространява се заедно с компилатора за микрокомпютрите Правец-16.

2. Богданов, Д., И. Мустакеров. Език за програмиране Си (с дискета). С., Техника, 1988 . Ще излезе от печат в началото на тази година.

От съветски автори:

3. Методические материалы и документация по пакетам прикладных программ. Вып. 46. МикроДОС. Язык программирования Си. М., МЦНТИ и МНИИСМ, 1986, 88 с.

Книги, преведени от английски на руски език:

4. Кернинган, Бл. Д. Ритчи, А. Фъюэр. Язык программирования Си. Задачи по языку программирования Си. М., Финансы и статистика, 1985, 279 с.

5. Хелмс, Г. Л. Языки программирования. Краткое руководство. М., Радио и связь, 1985, 175 с. Язык Си, с. 49-63.

6. Хэнкок, Л., М. Кригер. Введение в программирование на языке Си. М., Радио и связь, 1986, 193 с.

7. Болски, М. И. Язык программирования Си. Справочник. М., Радио и связь, 1988, 96 с.

8. Уэйт, М., С. Прата, Д. Мартин. Язык Си. Руководство для начинающих. М., Мир, 1988 , 512 с.

9. Берри, Р., Б. Микинс. Язык Си. Введение для программистов. М., Мир, 1988, 190 с. (Серия „Математическое обеспечение ЭВМ").

Книги на руски език по операционни системи, в които има раздели по езика Си:

10. Беляков, М. И. и др. Инструментальная мобильная операционная система ИНМОС. М., Финансы и статистика, 1985.

11. Кристиан, К. Введение в операционную систему UNIX. М., Финансы и статистика, 1985, 319 с. Глава 15. Язык Си и система UNIX с. 190—207.

12. Банахан. М., Э. Раттер. Введение в операционную систему UNIX. М., Радио и связь, 1986, 342 с. Глава 4. Язык программирования Си, с. 84—116.

13. Баурн, С. Операционная система UNIX. М., Мир, 1986, 464 с. (Серия „Математическое обеспечение ЭВМ").

14. Томас, Р., Дж. Йейтс. Операционная система UNIX. Руководство для ползователей. М., Радио и связь, 1986, 351 с.

АДА

От български автори:

1. Базисен програмен продукт Три/Ада. Ръководство за потребителя, 25 с. Описание на езика, 280 с. С., СО „Програмни продукти и системи", ТК „НППФ", 1988. Разпространява се заедно с компилатор за микрокомпютъра Правец-16.

Раденски, А. Програмиране на Ада. С., Наука и изкуство, 1989. Ще излезе от печат до края на първото полугодие на тази година.

Книги, преведени от английски на руски език:

3. Язык программирования Ада (предварительное описание). М., Финансы и статистика, 1981, 192 с.

4. Вегнер, П. Программирование на языке Ада. М., Мир, 1983, 240 с. (Серия „Математическое обеспечение ЭВМ").

5. Пайл, Я. Ада — язык встроенных систем. М., Мир, 1988, 320 с.

ТУРБО БЕЙСИК

1. Фирмена литература на английски от фирмата „Borland International Inc", която е създала езика TURBO BASIC (TB), заедно със система за програмиране на него, включваща компилатор от езика, редактор и др.

2. Иванов, Ч. Бейсик срещу Бейсик: Quick или Turbo? Компютър за вас, 1988, № 9—10, с. 20—28. В статията се сравняват езиците Куик Бейсик 3.0 на фирмата „Microsoft Corporation" и Турбо Бейсик на фирмата „Borland International Inc."

Редакцията на КВ не осигурява литература за читателите. Но за тези, които се интересуват по-отблизо от развитието на информатиката, най-разумно е да следят съветските бюлетини „Новые книги СССР" и да се абонират за съответните книги на руски. Това става всяка седмица чрез попълване на пощенски картички.

За българските издания можете да се абонирате само веднъж в годината в съответната книжарница за българска техническа литература. Използва се годишният каталог на ДИ „Техника". □

ЗА ВАШИЯ СПРАВОЧНИК

ОПИСАНИЕ НА КОМАНДИТЕ НА CP/M

Текстът, набран с получер шрифт, показва запазените думи, които трябва да се въвеждат така, както са написани. С курсив са изписани *думи, числа* или *букви,* които се въвеждат от потребителя.

Файлови имена

Файловете в CP/M се идентифицират с имена, които се състоят от две части:

►Задължително име на файл с не повече от осем знака;

►Незадължително разширение до три знака.

Ако се използва разширение, то се отделя от името на файла с точка. Както името, така и разширението могат да се състоят от главни букви, цифри и специални знаци с изключение на „<“ „>“, „,“, „;“, „:“, „=“, „*“, „?“, „[“, „]“ и интервал.

Валидни имена на файловете са например TEST.TXT, COST.JAN, GAMMA89.

Файлова спецификация

В CP/M има два вида файлова спецификация – еднозначна и неоднозначна.

►Еднозначната файлова спецификация фиксира името на точно определен файл, например PIP.COM.

►Неоднозначната файлова спецификация може да идентифицира и повече от един файл чрез използване на заместващи (wildcard) знакове. Знакът „?“ заменя произволен знак от името или разширението, например TE?T.ADD. HELLO.A??. Подобно е действието на знака „*“, който замества произволен низ с произволна дължина до допустимата съответно за името и/или разширението. Например файловата спецификация MYPROG*.DAT обединява действието на MYPROG.DAT, MYPROG?.DAT и MYPROG??.DAT. Трябва да се има предвид, че всички знакове, които стоят след „*“ от името или разширението, се игнорират, тоест спецификацията MYPROG.*T е еквивалентна на MYPROG.*.

Резидентни команди

Командите DIR и ERA допускат неоднозначни, докато REN, SAVE и TYPE изискват задължително еднозначни файлови спецификации.
d: Назначава устройство *d* за текущо.

DIR Показва списък на файловете в дискетата на текущото устройство.

DIR *d:* Показва списък на файловете в дискетата на устройство *d:*.

DIR *filename* Извежда списък на файловете на текущото устройство, които отговарят на зададената с *filename* файлова спецификация.

DIR *d:filename* Извежда списък на файловете на устройство *d:*, които отговарят на зададената с *filename* файлова спецификация.

ERA *filename* Изтрива файловете, отговарящи на спецификацията *filename* от текущото устройство.

ERA *d:filename* Изтрива файловете, отговарящи на спецификацията *filename* от устройство *d:*.

REN *old=new* Преименува файла с име *old* на *new* на текущото устройство.

REN *old=d:new* Преименува файла с име *old* от текущото устройство на *new* от устройство *d:*.

SAVE *n filename* Записва *n* 256-байтови страници от паметта във файла *filename* върху текущото устройство.

TYPE *d:filename* Показва на екран съдържанието на текстовия файл *filename* на устройство *d:*.

USER *n* Установява потребителска област $n, 0 \leqslant n \leqslant 15$.

Външни команди

Тези команди не стоят резидентно в паметта, а се прочитат от дискета. STAT допуска неоднозначни, а ASM, DDT, DUMP, LOAD и SUBMIT изискват еднозначни файлови спецификации.

ASM *filename* Компилира програма на асемблер от файла *filename* на текущото устройство и създава файловете *filename.HEX* и *filename.PRN*.

DDT *file.typ* Прави „дебъг“ на файла *file.typ* от текущото устройство. Ако не се посочи разширение, по подразбиране се търсят файлове с разширения .COM или .HEX.

DUMP *file.typ* Показва върху екран съдържанието на файла *file.typ* в шестнайсетичен код.

LOAD *filename* Създава изпълним файл *filename.COM* от сорсов файл на машинен език *filename.HEX*.

STAT Показва атрибутите и свободното дисково пространство на всички достъпни след последния старт дискетни устройства.

STAT *d:* Показва свободното дисково пространство на дискетата в устройство *d:*.

STAT *file* Показва размерите и атрибутите на файловете, отговарящи на спецификацията *file* от текущото устройство.

STAT *d:file* Показва размерите и

атрибутите на файловете, отговарящи на спецификацията *file* от устройство *d:*.

STAT *file* **$R/O** Назначава атрибут Read/Only (разрешава достъп до файла само за четене) на файловете, отговарящи на спецификацията *file* от текущото устройство.

STAT *d:file* **$R/W** Назначава атрибут Read/Write на файловете, отговарящи на спецификацията *file* от устройство *d:*

STAT *file* **$SYS** Назначава атрибут SYStem на файловете със спецификация *file*. Командата **DIR** не показва тези файлове.

STAT *file* **$DIR** Назначава атрибут DIRectory на файловете със спецификация *file*, с което елиминира назначения им по-рано атрибут **SYS**. Командата **DIR** показва тези файлове.

STAT DEV: Показва списък на текущите логически назначения на физическите устройства.

STAT VAL: Показва списък на възможните логически назначения на физическите устройства.

STAT CON:=TTY: Назначава физическото устройство **TTY** за логическо устройство на конзолата.

STAT *d:***DSK** Показва характеристиките на дискетата от устройство *d:*

STAT *d:***R/O** Назначава временно атрибут **R/O** на дискетата от устройство *d:*

SUBMIT *filename* Изпълнява последователност от CP/M-команди, записани в текстовия файл *filename.SUB*.

XSUB Разширява действието на командата **SUBMIT** като позволява вход от команден ред.

Командата PIP

Командата **PIP** (Peripheral interchange Program) се използва за копиране на файлове от едно системно устройство на друго.

PIP Командата **PIP** се изпълнява в интерактивен режим.

PIP *new=old* Копира файла *old* във файл *new* на текущото устройство.

PIP *c:=d:file* Копира файла *file* от дискетата в устройство *d:* на дискетата в устройство *c:* без да променя името.

PIP LST:=*file* Разпечатва на текущото печатащо устройство файла *file*.

PIP *c:new=d:old1.d:old2[0]* създава файл *new* на дискетата от устройство *c:*, който е обединение на файлове *old1* и *old2* от устройство *d:*.

Параметри на командата PIP

Параметрите на командата **PIP** се използват за промяна на действието ѝ. Параметрите се заграждат в средни скоби и се поставят в края на командния ред. Например командата **PIP** *COSTS.JAN=COSTS.DAT*[**EL**] копира файла *COSTS.DAT* във файл *COSTS.JAN*, като заменя всички главни букви с малки и показва файла на екрана.

B Копира се в блоков режим в буфер, докато се срещне символ ⟨CTRL–S⟩. След като се изкопират данните от буфера, продължава копирането на други данни.

D$_n$ Изтрива знаковете след колона *n*.

E Извежда копираната информация и върху екран.

F Изтрива от файла символите за нова страница.

G$_n$ Файлът се взема от потребителска област *n*.

H Проверява за коректност шестнайсетични данни.

! Проверява за коректност шестнайсетични данни. Записите „:OO“ се игнорират.

L Заменя главните букви с малки.

N Номерира редовете във файла.

N2 Номерира редовете във файла (водещите нули са включени), като след всеки номер се вмъква символ за табулация.

O Копира обектен файл.

P$_n$ Вмъква във файла символа за нова страница ⟨FF⟩ след всеки *n* реда. Ако *n*=1 или *n* е изпуснато, ⟨FF⟩ се вмъква след всеки 60 реда.

Q$_{string}$⟨**CTRL–Z**⟩ Спира копирането при срещане на знаков низ *string*.

S$_{string}$⟨**CTRL–Z**⟩ Копирането започва след срещане на знаков низ *string*.

R Копира (прочита) системни файлове.

T$_n$ Разширява табулацията до всяка *n*-та колона.

U Заменя малките букви с главни.

V Прави проверка за вярност при копиране.

W Прави запис върху файл с атрибут Read/Only.

Z Нулира бита за четност.

Контролни символи

⟨**CTRL–C**⟩ Рестартира CP/M.

⟨**CTRL–E**⟩ Премества курсора на следващия команден ред, без да се изпълни текущият.

⟨**CTRL–H**⟩ Изтрива знак преди курсора.

⟨**CTRL–J**⟩ Генерира ⟨LF⟩. Преминаване на нов ред.

⟨**CTRL–M**⟩ Изпълнява ⟨RETURN⟩.

⟨**CTRL–P**⟩ Копира извежданата на екрана информация на печатащото устройство до повторно натискане на ⟨**CTRL–P**⟩.

⟨**CTRL–R**⟩ Повтаря текущия ред.

⟨**CTRL–S**⟩ Прекъсва извеждането на информация на екрана до натискане на произволен клавиш.

⟨**CTRL–U**⟩ Изтрива текущия ред от паметта, но го запазва записан върху екрана. Курсорът се премества на следващия команден ред.

⟨**CTRL–X**⟩ Изтрива текущия ред от паметта и екрана.

⟨**CTRL–Z**⟩ Прекратява въвеждането от клавиатурата.

ОБЯВА

В бр. 7–8 на „КОМПЮТЪР ЗА ВАС“ поместихме визитната картичка на програмния продукт ГРАФ — ТЕКСТ 16. По този повод в редакцията писаха и се обадиха много читатели с молба да съобщим точния адрес на софтуерната къща „Правец — програма“.

По телефона научихме не само адреса, но и това, че продуктът се разпространява без ограничения.

А ето и визитната картичка на софтуерната къща в Правец:

2161 ПРАВЕЦ
Комбинат за микропроцесорна техника
Софтуерна къща „Правец — програма“
тел. 30-13 (код — 997133)

АТЛАС

ПОДПРОГРАМИ НА МОНИТОРА

ПЕТЪР ПЕТРОВ

Подпрограма BASCALC (**$FBC1**)
Извиква се, след като поредният номер на реда от екрана (0 — 23) се зареди в регистър А. Изчислява базовия адрес на този ред и го записва в клетки BASL, H (осъществява преобразуване на поредния номер на реда в адрес от екранната памет). Този адрес отговаря на първата позиция на реда от екрана и не зависи от параметрите на текстовия прозорец. Всички следващи адреси в рамките на същия ред се получават с индексна адресация по съдържанието на регистър Y. Подпрограмата BASCALC се извиква от почти всички подпрограми, използвани за управление на екрана в текстов режим. При нейното изпълнение се променя съдържанието на регистър А.

Подпрограма VTAB (**$FC22**)
Това е основната подпрограма за управление на движението на курсора. Извиква подпрограмата BASCALC и по съдържанието на клетки CV и WNDLFT изчислява базовия адрес. Записва получената стойност в клетки BASL, H и така установява вертикалната позиция на курсора. Обърнете внимание, че докато подпрограмата BASCALC изчислява абсолютния базов адрес, подпрограмата VTAB изчислява базовия адрес в рамките на текстовия прозорец. Ето защо подпрограмите, управляващи текстовия режим в рамките на текстовия прозорец, при изчисляване на базовия адрес не се обръщат директно към подпрограмата BASCALC, а използват подпрограмата VTAB. За целта, преди да се извика подпрограмата VTAB, в клетка CV се записва вертикалната позиция на курсора или директно се използва алтернативната входна точка VTABZ. Например:

```
LDA #$V     ;V = $0 - $17 - номер на ред от екрана
STA CV
JSR VTAB
```

Или:

```
LDA #$V
JSR VTABZ
```

В програмата МОНИТОР на Правец-8А не съществува подпрограма за установяване на хоризонталиата позиция на курсора. В зависимост от установения режим такава подпрограма се симулира по следните начини:

— в режим ТЕКСТ 40:

```
LDA #$H     ;H = $0 - $27 - хоризонтална позиция
STA CH      ;CH показва хоризонталната позиция
            ;на курсора (адрес $24)
```

— в режим ТЕКСТ 80:

```
DA #$H      ;H = $0 - $4F - хоризонтална позиция
STA $057B
```

При изпълнението на подпрограмата VTAB се променя съдържанието на регистър А.

Подпрограма VTABZ (**$FC24**)
Това е алтернативна входна точка на подпрограмата VTAB. По съдържанието на регистър А и на клетка WNDLFT изчислява базовия адрес и го записва в BASL, H. Не записва съдържанието на регистър А в клетка CV. Извиква се, след като номерът на реда от екрана се зареди в регистър А.

```
LDA #$V     ;V = $0 - $17
JSR VTABZ
```

При изпълнението на подпрограмата VTABZ се променя съдържанието на регистър А.

Подпрограма TABV (**$FB5B**)
И тази подпрограма е алтернативна входна точка на подпрограмата VTAB. Записва съдържанието на регистър А в клетка CV и безусловно предава управлението на подпрограмата VTAB. Извиква се, след като номерът на реда от екрана се зареди в регистър А. Изпълнението ѝ променя съдържанието на този регистър.

```
LDA #$V     ;V = $0 - $17
JSR TABV
```

Подпрограма UP (**$FC1A**)
Премества курсора с един ред нагоре, без да про

мения хоризонталната му позиция. Курсорът не се премества, ако се намира на най-горния ред от текстовия прозорец. Изпълнението на подпрограмата UP променя съдържанието на регистър А.

Подпрограма BS **($FC10)**

Подпрограмата BS премества курсора с една позиция наляво. Ако курсорът е в началото на реда, тя го премества в края на горния ред, като преизчисляването на вертикалната му позиция се извършва от подпрограмата UP.

Изпълнението на подпрограмата BS променя съдържанието на регистър А.

Подпрограмата BS не може да се използва в режим ТЕКСТ 80.

Подпрограма ADVANCE **($FBF4)**

Увеличава съдържанието на клетка CH с единица. Ако е достигнат краят на реда в рамките на текстовия прозорец, извиква подпрограмата CR.

При изпълнението на подпрограмата ADVANCE се променя съдържанието на регистър А.

Подпрограмата ADVANCE не може да се използва в режим ТЕКСТ 80.

Подпрограма STORADV **($FBF0)**

Подпрограмата STORADV, без да анализира съдържанието на регистър А, го записва на текущата позиция на курсора. Използва подпрограмата ADVANCE за преместване на курсора с една позиция надясно.

Изпълнението на подпрограмата STORADV променя съдържанието на регистри А и Y.

Подпрограмата STORADV не може да се използва в режим ТЕКСТ 80.

Подпрограма LF **($FC66)**

Премества курсора с един ред надолу, без да променя хоризонталната му позиция. За целта съдържанието на клетка CV се увеличава с единица и новата стойност се сравнява с долния край на текстовия прозорец. Ако (CV) > (WNDBTM), изображението на екрана се премества с един ред нагоре. Ако (CV) < (WNDBTM), извиква се подпрограмата VTABZ, която преизчислява новия адрес на курсора.

Изпълнението на подпрограмата LF разрушава съдържанието на регистър А, а при преместване на изображението с един ред нагоре се нарушава и съдържанието на регистър Y.

Подпрограма CR **($FC62)**

Премества курсора в началото на следващия ред от екрана. За целта записва нула в клетка CH (премества курсора в началото на текущия ред) и извиква подпрограмата LF, която премества курсора с един ред надолу.

Изпълнението на подпрограмата CR разрушава съдържанието на регистър А, а при преместване на изображението с един ред нагоре се разрушава и съдържанието на регистър Y.

Подпрограмата CR не може да се използва в режим ТЕКСТ 80.

Подпрограма SETTXT **($F839)**

Установява екрана в текстов режим и използва подпрограмата SETWND за нормализиране на параметрите на текстовия прозорец.

Изпълнението на подпрограмата SETTXT променя съдържанието на регистър А.

Подпрограма SETWND **($FB48)**

Подпрограмата SETWND нормализира размерите на текстовия прозорец. При изпълнението ѝ се променя съдържанието на регистър А.

Подпрограма SETIFLG **($FE86)**

Записва съдържанието на регистър Y в клетка INVFLG и така установява формата на изображението (НОРМАЛНО, ИНВЕРСНО или МИГАЩО ВИДЕО). Извиква се, след като стойността на маската за съответния формат се зареди в регистър Y.

```
$FF - НОРМАЛНО ВИДЕО
$7F - МИГАЩО ВИДЕО
$3F - ИНВЕРСНО ВИДЕО
```

Например:

```
LDY #$7F
JSR SETIFLG
```

Подпрограма SETNORM **($FE84)**

Установява формат НОРМАЛНО ВИДЕО. Използва подпрограмата SETIFLG за установяване на флаг INVFLG.

При изпълнението на подпрограмата SETNORM се променя съдържанието на регистър Y.

Подпрограма SETINV **($FE80)**

Установява формат ИНВЕРСНО ВИДЕО. Използва подпрограмата SETIFLG за установяване на флаг INVFLG.

При изпълнението на подпрограмата SETNORM се променя съдържанието на регистър Y.

Подпрограма INIT **($FB2F)**

Нулира съдържанието на клетка STATUS (адрес $48). Изключва режим ГР280. Нулира програмноуправляемия ключ СТР2. Установява текстов режим. Установява нормалните размери на текстовия прозорец.

При изпълнението на подпрограмата INIT се променя съдържанието на регистър А.

Подпрограма CLREOL **($FC9C)**

Изчиства реда от текущата позиция на курсора до десния край на текстовия прозорец. Позицията на курсора се определя от хоризонталната му координата и от базовия адрес BASL, H.

Изпълнението на подпрограмата CLREOL разрушава съдържанието на регистри А и Y, а курсорът запазва позицията си.

Подпрограмата CLEOLZ **($FC9E)**

Изчиства текущия ред от екрана. Започва от позицията, определена от базовия адрес BASL, H и съдържанието на регистър Y, и завършва до десния край на текстовия прозорец. Извиква се, след като в регистър Y се посочи позицията, откъдето трябва да започне изтриването. Например:

```
LDY H      ;H = $0 - $24 за 40 колони
           ;H = $0 - $4F за 80 колони
JSR CLEOLZ
```

Изпълнението на подпрограмата CLEOLZ разрушава съдържанието на регистри A и Y, а курсорът запазва позицията си.

Подпрограма CLREOP **($FC42)**

Изчиства екрана от текущата позиция на курсора до края на текстовия прозорец. Изпълнението й разрушава съдържанието на регистри A и Y, а курсорът запазва позицията си.

Подпрограма HOME **($FC58)**

Позиционира курсора в началото на текстовия прозорец (горния му ляв край) и изчиства екрана. За целта нулира клетките, съдържащи хоризонталната и вертикалната координата на курсора, и извиква подпрограмата CLREOP.

Изпълнението на подпрограмата HOME променя съдържанието на регистри A и Y.

Подпрограма SCROLL **($FC70)**

Премества изображението, ограничено от рамките на текстовия прозорец, с един ред нагоре. Съдържанието на най-горния ред от текстовия прозорец се губи, а най-долният ред се изчиства.

Изпълнението на подпрограмата SCROLL разрушава съдържанието на регистри A и Y, а курсорът запазва позицията си.

Подпрограма SCROLLDN **($CCAA)**

Премества изображението, намиращо се в текстовия прозорец, с един ред надолу. Изпълнява се само под управление на програмното осигуряване, поддържащо 80-колонно изображение. Например:

```
STA $C007     ;Разрешава достъпа до резидентната
              ;постоянна памет на адреси $C100 -
              ;$CFFF
JSR SCROLLDN
STA $C006     ;Разрешава достъпа до постоянната
              ;памет от областта $C100 - $CFFF,
              ;разположена на допълнителните модули
```

Подпрограми за поддържане на графика с ниска разделителна способност (режим ГР40)

Подпрограмите за поддържане на изображението в режим ГР40 са много близки до съответните команди в Бейсик. Те включват подпрограми за установяване на цвят, за изчистване на целия екран или част от него, за изобразяване на блокче, на хоризонтална и на вертикална линия в текущия цвят и за определяне на цвета на зададено блокче от екрана. Към тях условно се причисляват и подпрограмите за установяване на режим ГР40 и за нормализиране на текстовия прозорец. Не е трудно да се установи, че основна е подпрограмата от най-ниско ниво GBASCALC, която изчислява базовия адрес за всеки ред от графичното изображение (адреса на клетката от екранната памет, отговаряща на началото на съответния ред от екрана). Подпрограмите NXTCOL и SETCOL, които установяват текущия цвят, не са свързани с останалите.

Подпрограма SETGR **($FB40)**

Започва от адрес $FB40 и последователно изпълнява следните функции:

1. Установява графичен режим. Обърнете внимание, че подпрограмата SETGR не установява нито вида на графичния режим (ГР40 или ГР280), нито графичната страница. И графичният режим, и страницата трябва да са установени предварително.

2. Установява режим на смесено изображение.

3. Извиква подпрограмата CLRTOP, която изчиства първите 40 реда от екрана в режим ГР40. Следователно, ако подпрограмата SETGR бъде извикана след установяването на режим ГР280, тя няма да изчисти съответната графична страница.

4. Преминава в подпрограма SETWND.

За установяването на целия екран в режим ГР40 няма готова подпрограма, но такава може да се симулира по следния начин:

```
LDA $C050     ;Установява графичен режим
LDA $C052     ;Забранява режима на смесено
              ;изображение
LDA $C056     ;Установява графика с малка
              ;разделителна способност
LDA $C054     ;Избира първа страница
```

Изпълнението на подпрограмата SETGR променя съдържанието на регистри A и Y.

Подпрограма NEXTCOL **($F85F)**

Извлича кода на текущия цвят за режим ГР40 от клетка COLOR (адрес $30) и го увеличава с три, след което преминава в подпрограма SETCOL.

```
COLOR = (COLOR + 3) MOD 16
```

При изпълнението на подпрограмата NEXTCOL се променя съдържанието на регистър A.

Подпрограма SETCOL **($F864)**

Установява текущия цвят в режим ГР40. Преди нейното извикване е необходимо кодът на цвета да се зареди в регистър A (A = 0 ÷ 15). Подпрограмата SETCOL преобразува този код така, че двете половинки на байта да са еднакви, и го записва в клетка COLOR. Така например след изпълнение на следната програма на адрес $30 има $66:

```
LDA #$06
JSR SETCOL
```

Изпълнението на подпрограмата SETCOL води до промяна на съдържанието на регистър A.

Подпрограма GBASCALC **(**$F847**)**

Преобразува координатите на блокче от екрана в адрес от екранната памет. Извиква се, след като вертикалната координата на блокчето се зареди в регистър A (A = 0 ÷ 23). Изчислява и записва в GBASL, H (адреси $38 и $39) базовия адрес. Това е адресът на клетката от екранната памет, отговарящ на най-лявото блокче от съответния ред на екрана. Използва се от почти всички подпрограми, свързани с режим ГР40. При изпълнението на подпрограмата GBASCALC се променя съдържанието на регистър A.

Подпрограма PLOT **(**$F800**)**

Изчертава единично блокче в установения цвят. Извиква се, след като хоризонталната и вертикалната координата на блокчето са заредени в регистри Y (Y = 0 ÷ 39) и A (A = 0 ÷ 47). Използва подпрограмата GBASCALC за изчисляване на базовия адрес на съответния ред от екрана. Определя дали блокчето попада на четен, или на нечетен ред от екрана и в зависимост от това записва $0F или $F0 в клетка MASK (адрес $2E). При изпълнението ѝ се променя съдържанието на регистър A.

Подпрограма HLINE **(**$F819**)**

Използва се за изчертаване на хоризонтална права линия с определен цвят. Преди да се извика подпрограмата HLINE, е необходимо желаният цвят да се установи с подпрограмата SETCOL, да се запише хоризонталната координата на крайната точка на правата в клетка H2 (адрес $2C) и да се заредят регистри A и Y с вертикалната координата на правата и с хоризонталната координата на началната ѝ точка.

Например:

```
LDA #$04      ;Изчертай с тъмнозелен цвят
JSR SETCOL    ;
LDA #$12      ;хоризонтална права линия от
STA H2        ;колона 0 до колона 18
LDA #$02      ;на ред 2
LDY #$00
JSR HLINE
```

При изпълнението на подпрограмата HLINE се променя съдържанието на регистър A.

Подпрограма VLINE **(**$F828**)**

Използва се за изчертаване на вертикална права линия в предварително установен цвят. Преди да се извика подпрограмата VLINE, е необходимо да се установи желаният цвят, да се запише вертикалната координата на крайната точка на правата в клетка V2 (адрес $2D) и да се заредят регистри A и Y с вертикалната координата на началната точка и хоризонталната координата на правата.

При изпълнението на подпрограмата VLINE се променя съдържанието на регистър A.

Подпрограма CLRSCR **(**$F832**)**

Това е подпрограма за изчистване на екрана в режим ГР40. Може да се разглежда като отделна входна точка на обща подпрограма за изчистване на екрана. В зависимост от входната точка тази обща подпрограма запълва целия екран или част от него с код $00 (в режим ГР40 този код отговаря на черен цвят, а в текстов режим — на знака C във формат ИНВЕРСНО ВИДЕО). Запълването на екрана се извършва колона по колона с подпрограмата VLINE. При изпълнението на коя да е от подпрограмите за изчистване на екрана в режим ГР40 се променя съдържанието на регистри A и Y.

Подпрограмата CLRSCR изчиства целия екран (48 реда).

Подпрограма CLRTOP (**$F836**)

Тя е алтернативна входна точка на програмата за изчистване на екрана. Използва се в режим ГР40 за изчистване на първите 40 реда от екрана.

Подпрограма CLRSC2 **(**$F838**)**

Тази подпрограма също е алтернативна входна точка на програмата за изчистване на екрана. Изчиства екрана от най-горния ред до реда, определен от съдържанието на регистър Y.

Подпрограма CLRSC3 **(**$F83C**)**

И тя е алтернативна входна точка на програмата за изчистване на екрана. Изчиства горния ляв ъгъл на екрана. Започва от точката с координати H = V = 0 и продължава до реда и колоната, определени от съдържанието на регистър Y и на клетка V2.

Подпрограма SCRN **(**$F871**)**

Използва се за определяне на цвета на единичното блокче с координати, зададени със съдържанието на регистри A и Y. Извиква се, след като вертикалната и хоризонталната координата на блокчето се заредят в регистри A и Y. След завършване на подпрограмата SCRN кодът на цвета на блокчето е записан в регистър A.

КВ. [illegible] 89

Инж. ПЕТЪР ХРИСТОВ
Н. с. инж. НИКОЛАЙ НИКОЛОВ

COMPILER PLUS

Компилаторът е програмно средство за „превеждане“ на потребителски програми, написани на език от високо равнище, във файлове на машинен код. Вграденият Бейсик — интерпретатор превежда програмните редове един след друг по време на изпълнение, което значително намалява бързодействието. Компилаторът транслира предварително програмата, след което се стартира изпълнимият код.

Съществуват много компилатори от разширен Бейсик за Правец-82 — SPEEDSTAR, TASC, БЕЙСИК-КОМПИЛАТОР на СО „Програмни продукти и системи“ и др. В сравнение с посочените компилатори работата с COMPILER PLUS е пределно опростена и достъпна за неспециалиста.

Предлагаме едно съкратено упътване за ползването му, което е систематизирано извлечение от оригиналното ръководство с обем 44 страници.

ПОДГОТОВКА НА ПРОГРАМАТА ЗА КОМПИЛИРАНЕ

* Преди да се пристъпи към компилация, програмата трябва да се въведе в оперативната памет на компютъра.

* Програмата се стартира от интерпретатора и се настройва неколкократно, по възможност с различни входни данни. Към компилиране се преминава само когато програмата е изчистена от грешки и работи безотказно.

* Проверява се дали програмата е съобразена с особеностите на COMPILER PLUS. Трябва да се има предвид и информацията за разположението на компилираната програма в оперативната памет.

ОСОБЕНОСТИ НА COMPILER PLUS

1. Забранено е използването на операторите:

DEL, LIST, LOAD, SAVE, LOMEM, TRACE, NOTRACE, STORE, RECALL, SHLOAD.

Употребата им ще доведе до грешка в процеса на компилиране. Изключение прави само операторът LOMEM: — той се пренебрегва.

2. При деклариране на масивите чрез оператора DIM размерностите им трябва да са константи. Не е допустимо използването на формули или променливи. Причината за това ограничение се крие в принципа на работа на компилатора. Докато интерпретаторът отделя памет за променливите динамично по време на изпълнение, компилаторът я определя статично — преди стартиране на самата програма. Това също повишава бързодействието на програмата.

3. Разширената структура на изходната бейсикова програма включва файл с главната програма и файлове — подпрограми, които ще наричаме програмни модули. Всяка подпрограма може да има собствени подпрограми и т. н. Програмните модули, без главната програма, съставят тъй наречената външна библиотека. Всички файлове се компилират поотделно.

РАЗПОЛОЖЕНИЕ НА КОМПИЛИРАНАТА ПРОГРАМА В ПАМЕТТА

Компютърът може да адресира до 65536 байта оперативна памет — адреси от 0 до 65535 ($0000 до $FFFF в 16-тичен запис). По принцип компилираната програма може да се разположи на произволно място в паметта, но на практика се използва пространството между адреси $0803 и $95FF. Останалата памет е запазена за ДОС — екранните буфери, нулевата страница и др.

Компилираната програма се състои от четири главни области:

* Област B (основна) — съдържа самата програма: кодове, литерали, константи, дефинирани функции и др. Тази област се запазва на дискета като изходен двоичен файл от работата на компилатора;

* Област L (LOMEM:) — адресно пространство, запазено за скаларните променливи и масивите;

* Област S (низове) — за съхраняване на текстовите (знаковите) променливи;

* Област X (външна) — тук се разполага служебният файл LIB.MOD с дължина 3.3 Кбайта, включен също в библиотеката. Този файл се използва при обръщение към програмен модул.

* Областите B, L и X са статични — не нарастват по време на изпълнението на програмата. Областта S нараства, тъй като старите стойности на текстовите променливи не се изтриват автоматично.

ПОДГОТОВКА ЗА КОМПИЛИРАНЕ

1. При необходимост като първи програмен ред във файла се добавя:

nnnnn REM! B$××××, L$××××, X$××××, S$××××, C$××××, A, J0, J1, J2, J3, D, M

Тук

nnnnn е номерът на програмния ред;

×××× е съответният шестнайсетичен адрес на паметта.

Празните знакове се игнорират от компилатора, а параметрите могат да се запишат в произволен ред. За интерпретатора този допълнителен ред е само коментар, който не пречи при изпълнението на програмата.

Първите пет параметъра служат за разпределение на оперативната памет на компютъра:

B$×××× — начален адрес на програмата в оперативната памет (начало на областта B);

L$×××× — начален адрес за разполагане на простите променливи (LOMEM:);

X$×××× — начален адрес на помощната програма LIB. MOD, необходима при извикване на файл — подпрограма. Тя може да бъде заредена, като в главната програма се включи програмният ред

nnnnn PRINT CHR$(4); „BLOAD LIB.MOD,A$××××“ преди извикването на компилираната подпрограма чрез

nnnnn PRINT CHR$(4); „BLOAD модул“

nnnnn CALL *адрес*

Друг начин е да се използва помощната програма MAKE LOADER (виж по-надолу). LIB. MOD се зарежда еднократно. X$ и A$ ползват един и същ адрес;

S$×××× — начален (минимален) адрес за разполагане на текстовите (знаковите) променливи, начало на областта S;

C$×××× — краен (максимален) адрес на текстовите променливи (HIMEM:).

Ако тези параметри не се зададат, компилаторът прилага стандартната конфигурация за разпределение на паметта, в която отделните области следват една след друга — B, L и S, започвайки от адрес $0803 (2051).

При стартиране главната програма изпълнява RETURN към печатащото устройство, CLEAR — към входа, и RESET — към изхода си, а другите програмни модули — евентуално само CLEAR. Всеки от следващите пет параметъра характеризира даден тип файл — подпрограма:

* Тип A — подпрограмата се извиква от некомпилирана главна програма. Автоматично се изпълнява CLEAR.

Бейсиковата програма може да извика повече от една подпрограма от тип A, но не може да извика подпрограма от тип J. Модулите от тип J могат да бъдат активирани от тялото на подпрограма тип A.

* Тип J0 — подпрограмата няма собствени променливи и DATA-данни. Използват се променливите и данните на извикващата програма, потиска се CLEAR. Това е най-простият вид подпрограма.

* Тип J1 — също като J0, но има собствени DATA-данни. Операторите READ четат данни само от собствените оператори DATA. При повторно извикване данните се четат по-нататък.

* Тип J2 — подпрограмата има собствени променливи, но няма собствени DATA-данни. CLEAR се потиска, но може да се използва подходящо в този модул по преценка на програмиста;

* Тип J3 — подпрограмата има както собствени променливи, така и DATA-данни. За CLEAR е в сила казаното за случая J2.

Останалите параметри обхващат специални случаи:

* G — автоматично изчиства излишните низове от областта на текстовите променливи при опасност от препълване. Това намалява бързодействието, тъй като при всяко присвояване на нова стойност от текстова променлива се проверява дали областта S не е препълнена. Затова вместо параметъра G се предпочита периодично използване на оператора FRE (×);

* D — забранява на ДОС да спира изпълнението на програмата при установена грешка във файлова операция. Действа само ако програмата не предвижда обработка на грешките (липсва оператор ONERR GOTO);

* M — информира компилатора, че ДОС ще бъде разположен на 16-Кбайтова рам-карта, поставена в слот 0. Преместването се осъществява след стартиране на служебния файл MMS (съкращение от Memory Management System) непосредствено след зареждане на ДОС. Освободеното допълнително място в паметта от адрес $9600 до $BFFF може да се използва за разполагане на компилираната програма. [illegible]ването на параметър D [illegible]ля употребата на M.

Параметрите на [illegible] то могат да не се [illegible] работка на програма, [illegible] стои само от [illegible]

2. Когато [illegible] ла обменят [illegible] менливи, те трябва да използват една и съща L-област, т. е. трябва да са компилирани с еднакви параметри L. Тогава всички техни променливи и масиви се обявяват по един и същ начин. За целта като втори и трети програмен ред и в двата файла се включват:

nnnnn REM *списък на всички променливи*

nnnnn DIM *списък на всички масиви*

Това е най-удобният начин за прехвърляне на данни между даден модул и негова подпрограма. Подпрограмата задължително трябва да бъде от тип J0 или J1. Освен това тя автоматично използва областта S на извикващия модул, като компилаторът пренебрегва параметрите ѝ S и C, ако са зададени.

КОМПИЛИРАНЕ

Програмата BASIC се компилира файл по файл в следната последователност:

1. Файлът се зарежда в паметта: LOAD *име на файл*.

2. Стартира се компилаторът: BRUN COMPILER. След кратко зареждане от дискетата на екрана се появява потвърждаващ надпис и компилирането започва. Можете да поискате кратки съобщения за фазата на компилиране, като напечатате „?“. В [illegible] случай до завършване на компилирането на екрана не се извежда нова информация.

Ако [illegible] открие грешка, например [illegible] записан оператор, се [illegible] съобщение за вида на грешката и номера на програмния ред, в който е открита. След прекъсване на ДОС от [illegible], управлението се предава на [illegible]ратора, но бейсиковата програма е окончателно изтрита от паметта. Затова никога не [illegible] програма, която не е предварително запазена на [illegible]! Работата на компилатора винаги може да се прекъсне чрез RESET.

3. След завършване на компилирането на екрана се извежда следното меню:

КВ. 03-04 89

0 TO DISPLAY MAP ON SCREEN
1—7 TO ROUTE TO SLOT
<RETURN> TO SAVE FILE
<ESC> TO EXIT

Изборът на

0 — показва картата с 16-ичните адреси на компилираните части на обработения файл;

1—7 — извежда картата към съответния слот;

<RETURN> — записва резултантния двоичен файл на дискета;

<ESC> — завършва работата на компилатора.

След натискане на RETURN от дискетата с компилатора се чете информация (не я вадете от дисковото устройство) и се извежда въпрос за името на компилирания файл.

Ако е възникнал конфликт при разпределението на паметта, на съответното място в картата 16-ичният адрес се появява знакът „?" вместо „$".

Нека разгледаме един пример. Да допуснем, че след натискане на 0, на екрана се вижда:

START BYTES SECTION
--- --- ---
$4000 $0006 MODULE ENTRY
$4006 $0215 LITERALS
$421B $01AE CONSTANTS
$43C9 $47BE MAIN CODE
$7B05 $00AE ARRAY VECTORS
$7BB3 $08C3 RUN-TIME MODULES
$8476 --- END BIN FILE+1
$8476 $0161 SCALER SPACE
$85D7 $15B0 ARRAY SPACE
$9B87 ?FA79 STRING POOL
$9600 --- INITIAL HIMEM

Началният адрес на компилираната програма — $4000, показва, че е бил използван параметърът B$4000. Най-често по този начин се предпазва съдържанието на първата графична страница $2000—$4000, когато програмата включва графика с висока разделителна способност. Стойността на END BIN FILE+1 (край на двоичния файл+1) сочи, че целият двоичен файл (областта B) се разполага в паметта преди ДОС. Това прави възможно повторното му зареждане. Картата показва също, че при стартирането на програмата областта на променливите — L, и областта на низовете — S, ще припокрият ДОС. Ако програмата не използва ДОС, тази конфигурация на паметта е допустима, въпреки че впоследствие ДОС трябва да бъде презаредена. За тази цел служи програмният ред

nnnnn POKE 977,166: POKE 978,250

Ако програмата използва ДОС, все още можете да опитате да я поберете в оперативната памет. При едно по-внимателно разглеждане на паметта се забелязва, че областта от $800 до $2000 стои неизползвана. Логично е да се постави там областта L чрез промяна на командния ред, например

10 REM! B$4000, L$800

По този начин проблемът се решава. Когато програмата е прекалено голяма, за да се побере в паметта, най-доброто решение е да се раздели програмата на два или повече програмни модула. Тези модули се компилират с еднакъв начален адрес (параметър B) и всеки от тях заема едно и също пространство в паметта, когато е зареден. Тогава казваме, че модулите „се припокриват". Едно типично приложение на тази техника би могла да бъде програма — меню, съставена от една главна програма — резидентна в паметта, която управлява зареждането на другите програмни модули и обмяната на информация между тях.

РЕЗУЛТАТ

Компилираният по премълчаване файл (без зададени параметри) се стартира от дискетата чрез BRUN или BLOAD и CALL 2051. Ако е зададен параметърът B$××××, неговият десетичен еквивалент замества 2051 в оператора CALL.

Ако програмата е съставена от няколко файла, се стартира главната програма. При компилирана главна програма, в която е пропусиат програмен ред за зареждане на LIB.MOD, може да се използва файлът MAKE LOADER.

След стартирането му (RUN MAKE LOADER) на екрана се извеждат последователно въпроси за името на главната програма, началния адрес на разположение на LIB.MOD в паметта и имената на използваните подпрограми. Тази информация се запазва на дискетата във вид на нов двоичен файл, при стартирането на който се зареждат в паметта всички посочени програмни модули заедно с файла LIB.MOD, след което се стартира главната програма.

КОМПИЛИРАНАТА ПРОГРАМА ВСЕ ПАК МОЖЕ И ДА НЕ РАБОТИ!

Причината вероятно се крие в неправилното разположение на данните в оперативната памет на компютъра. Не трябва да се забравя, че картата с адресите се отнася само за един файл и е възможно неправилно припокриване на програмните модули. Но това се случва рядко, при големи и сложни програми. С малко повече внимание всичко може да се оправи.

В заключение трябва да отбележим, че по-голяма част от програмите се компилират бързо и без всякакви трудности.

□

ОБЯВА

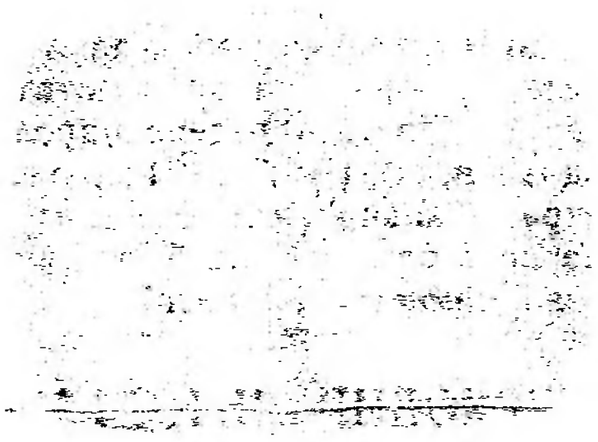
Притежавам домашен компютър Правец–8Д и разполагам с доста програми, някои от които съм съставил сам. Умея да програмирам на Бейсик и Асемблер. Желая контакти с други притежатели на Правец–8Д с цел размяна на програми и обмен на опит и литература. Освен това търся желаещи за съвместна работа — разработване на големи програмни продукти.

БАЛЧИК
Варненска област
ул. Г. Бенковски 5
Красимир Костов

РАБОТА С LOCKSMITH 6.0

Помощни програми към DOS 3.3

(DOS 3.3 UTILITIES)



Инж. ЙОРДАН ЙОРДАНОВ

Под това название са обединени няколко програми за работа с дискети. Функцията се задейства с натискане на клавиша < D > от главното меню на системата. На екрана се изписва списъкът от програми, включени във функцията.

Всяка функция се избира с натискане на съответстващия ѝ клавиш.

СПРАВОЧНИК НА ДИСКА (< C > — CATALOG)

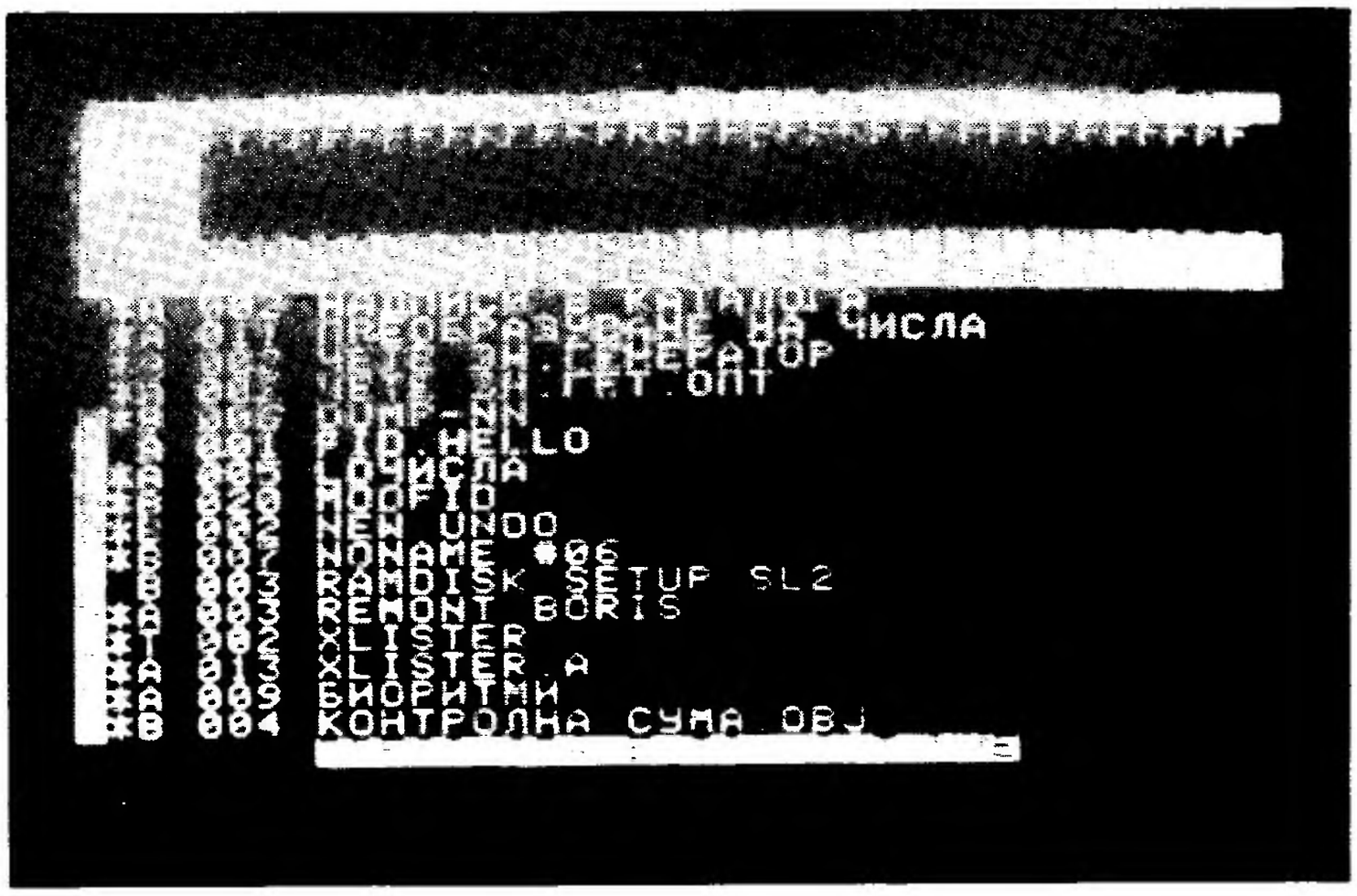
С натискане на клавиша < C > се изписва справочникът на диска. Изтритите файлове са маркирани с D.

КОРИГИРАНЕ НА СПРАВОЧНИКА НА ДИСКА (< K > — FIX SECTOR COUNTS)

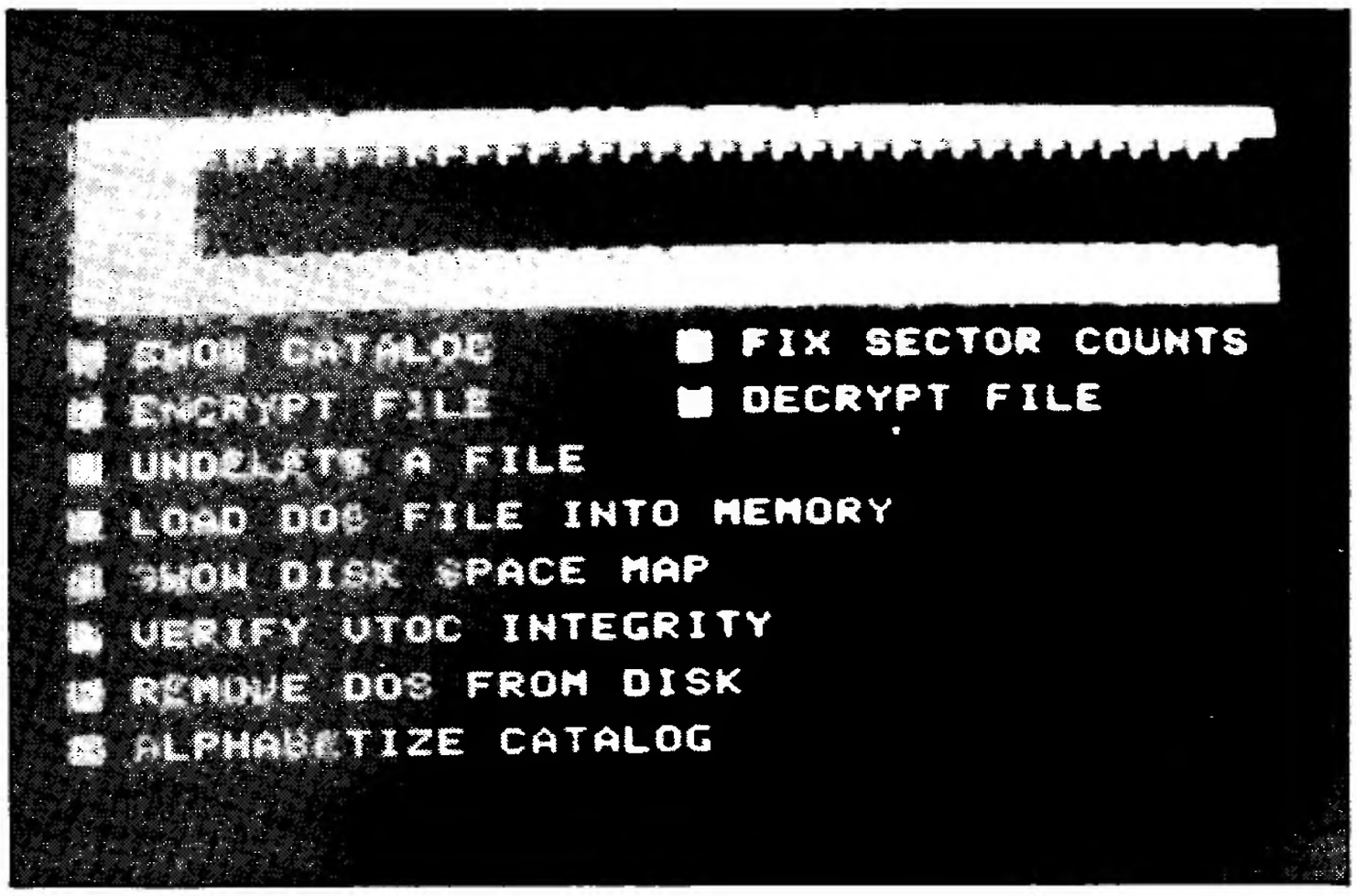
При някои случаи ДОС записва в каталога по-голям от действително заетия от файла брой сектори. Чрез натискане на клавиша < K > се задейства функцията, която коригира показанията в каталога. След това задава въпрос, дали да обнови съдържанието на справочника. Ако се натисне клавишът за интервал, справочникът се обновява. С клавиша ESC обновяването се отказва.

КОДИРАНЕ НА ФАЙЛ (< E > — ENCRYPT FILE)

След натискане на клавиша < E > системата запитва за номера на устройството и за името на файла, който ще се кодира. След въвеждане на името се проверява дали файлът съществува. Ако файлът е намерен, трябва да въве-



Фиг. 2

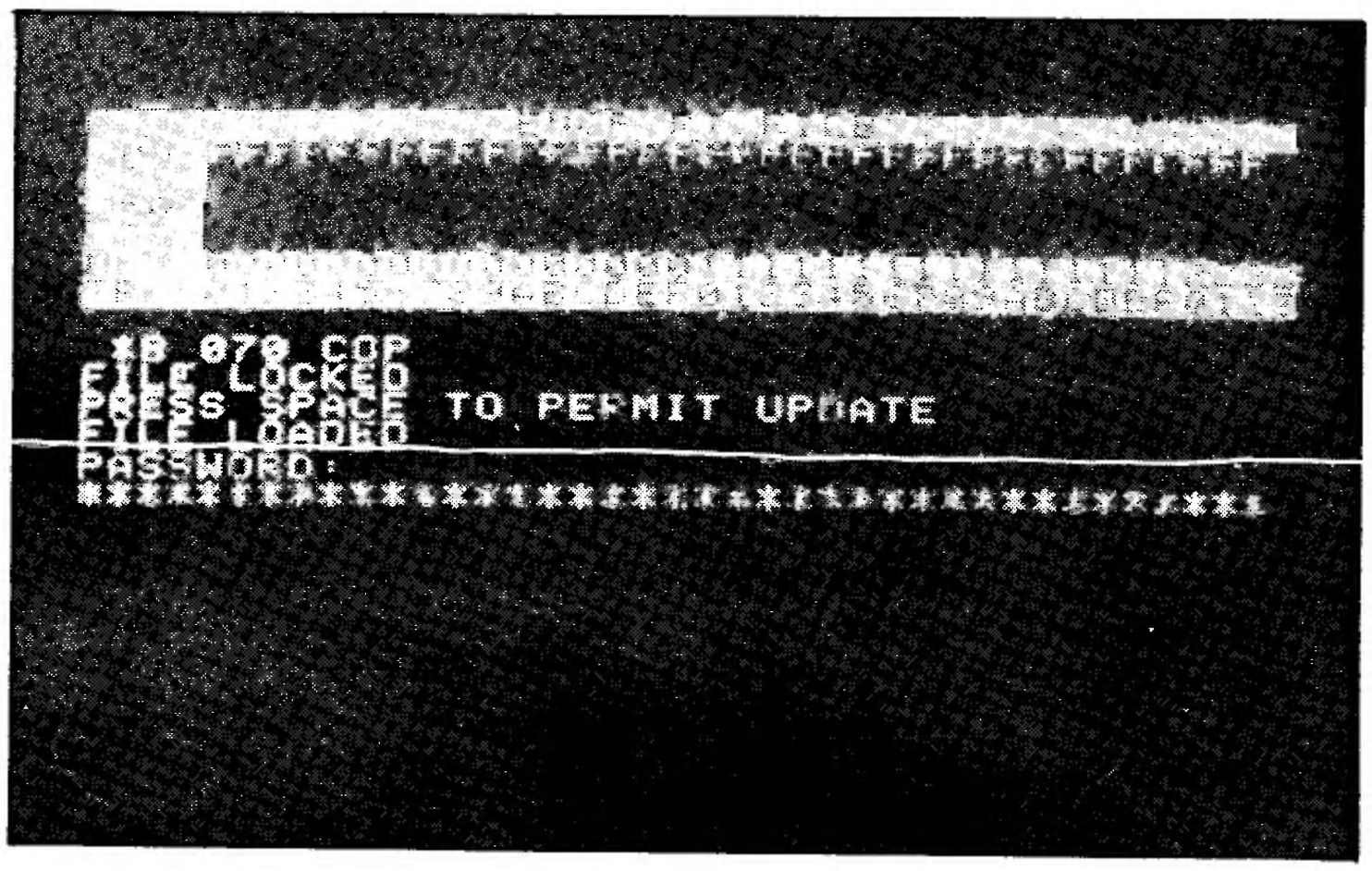


Фиг. 1

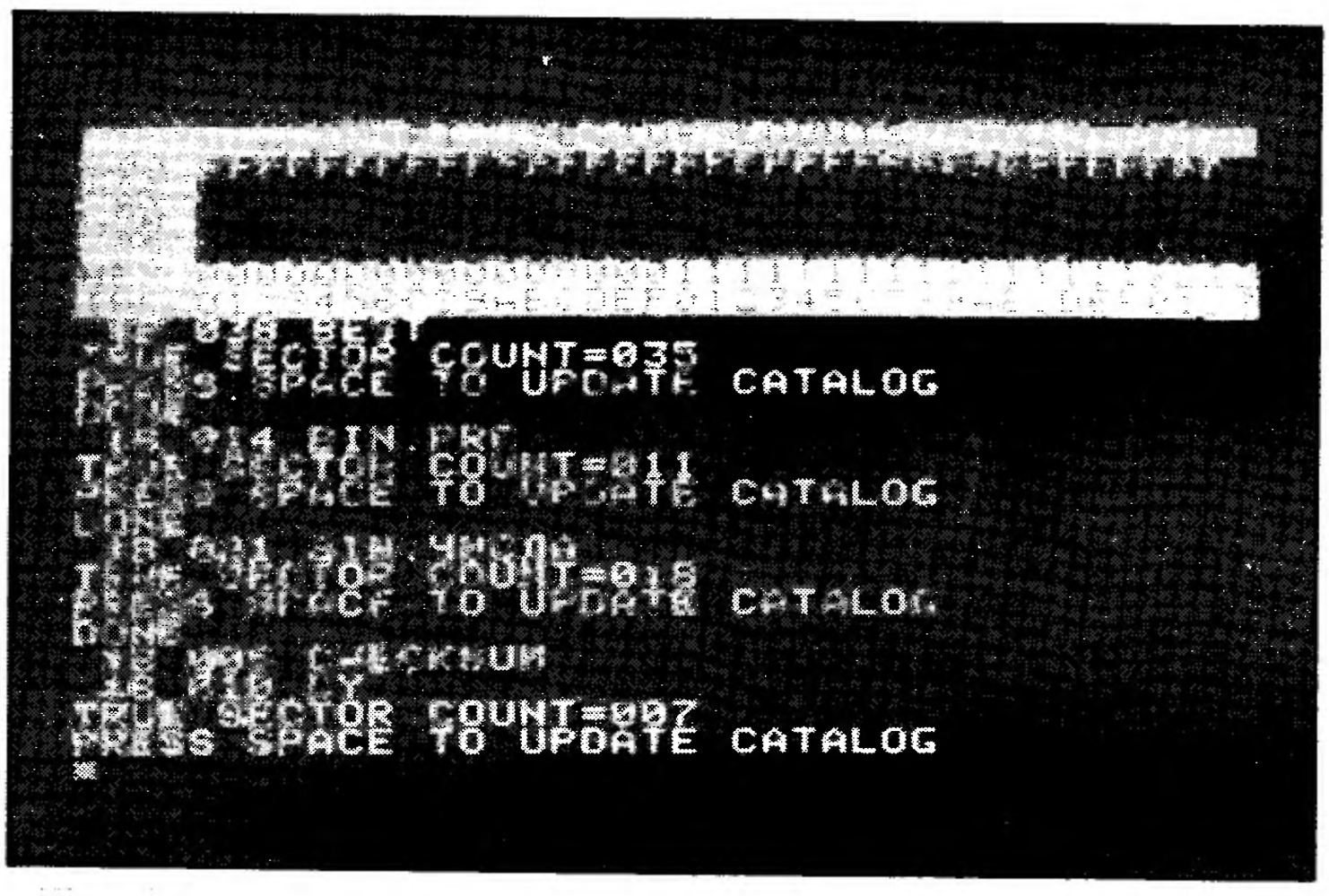
дете парола за кодирането. При въвеждане паролата не се изписва на екрана и трябва да я въведете повторно, за да я потвърдите. Ако двете въведени пароли са еднакви, файлът се кодира. На екрана кодираният файл изглежда като безсмислена бъркотия, затова **помнете паролата!**

ДЕКОДИРАНЕ НА ФАЙЛ (< D > — DECRYPT FILE)

Тази функция е обратна на предходната. Тя се задейства с натискане на клавиша < D > от списъка програми. Ако файлът е кодиран два пъти, например от двама души, всеки от които е използвал собствена парола, той се декодира в обратен ред: първо с втората парола и след това с първата. Кодираните файлове трябва непременно да се копират на втора дискета, защото, ако бъде направен опит да се декодират с грешна парола, те се объркват още повече.



Фиг. 4



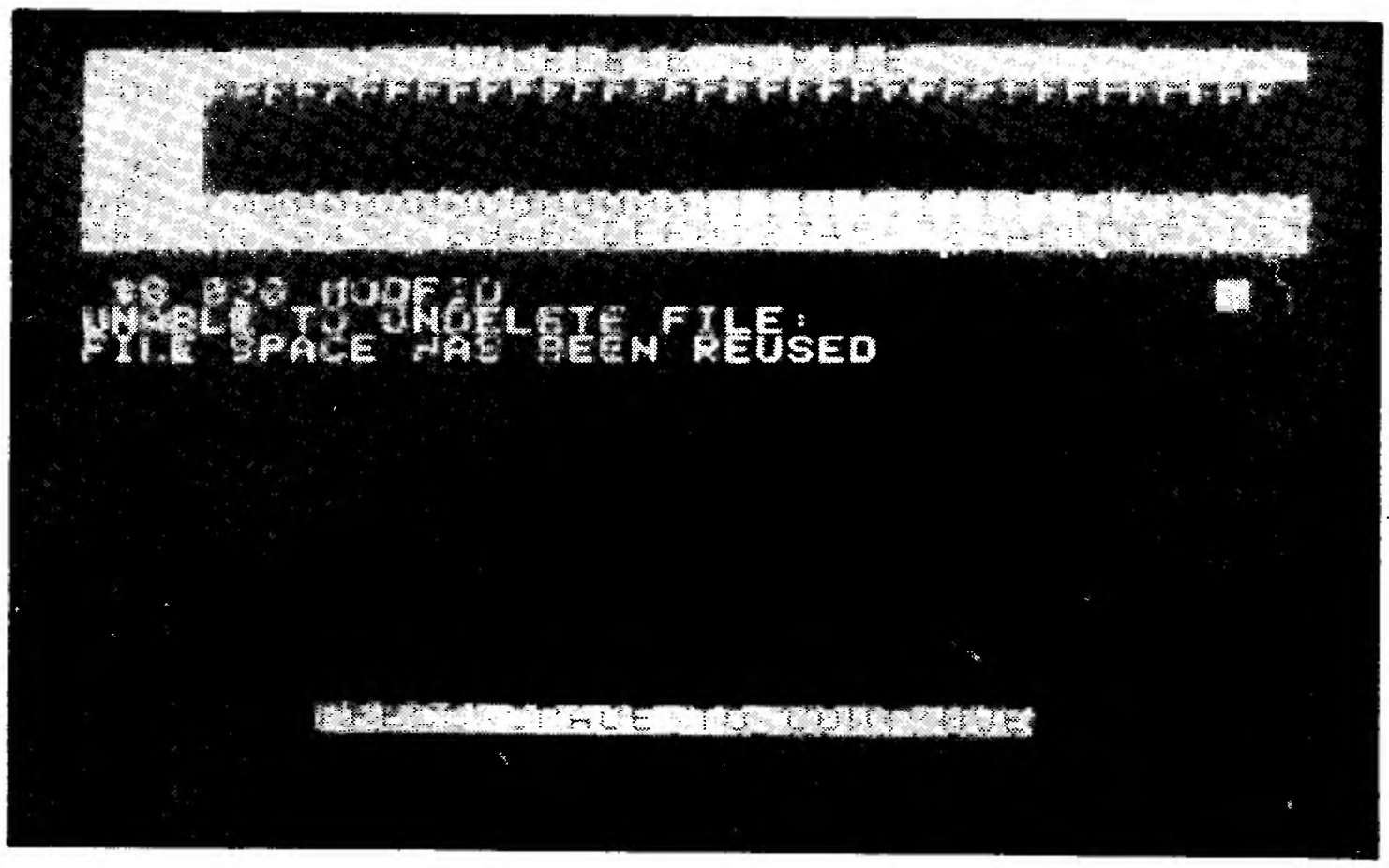
Фиг. 3

ВЪЗСТАНОВЯВАНЕ НА ИЗТРИТ ФАЙЛ (< U > — UNDELETE A FILE)

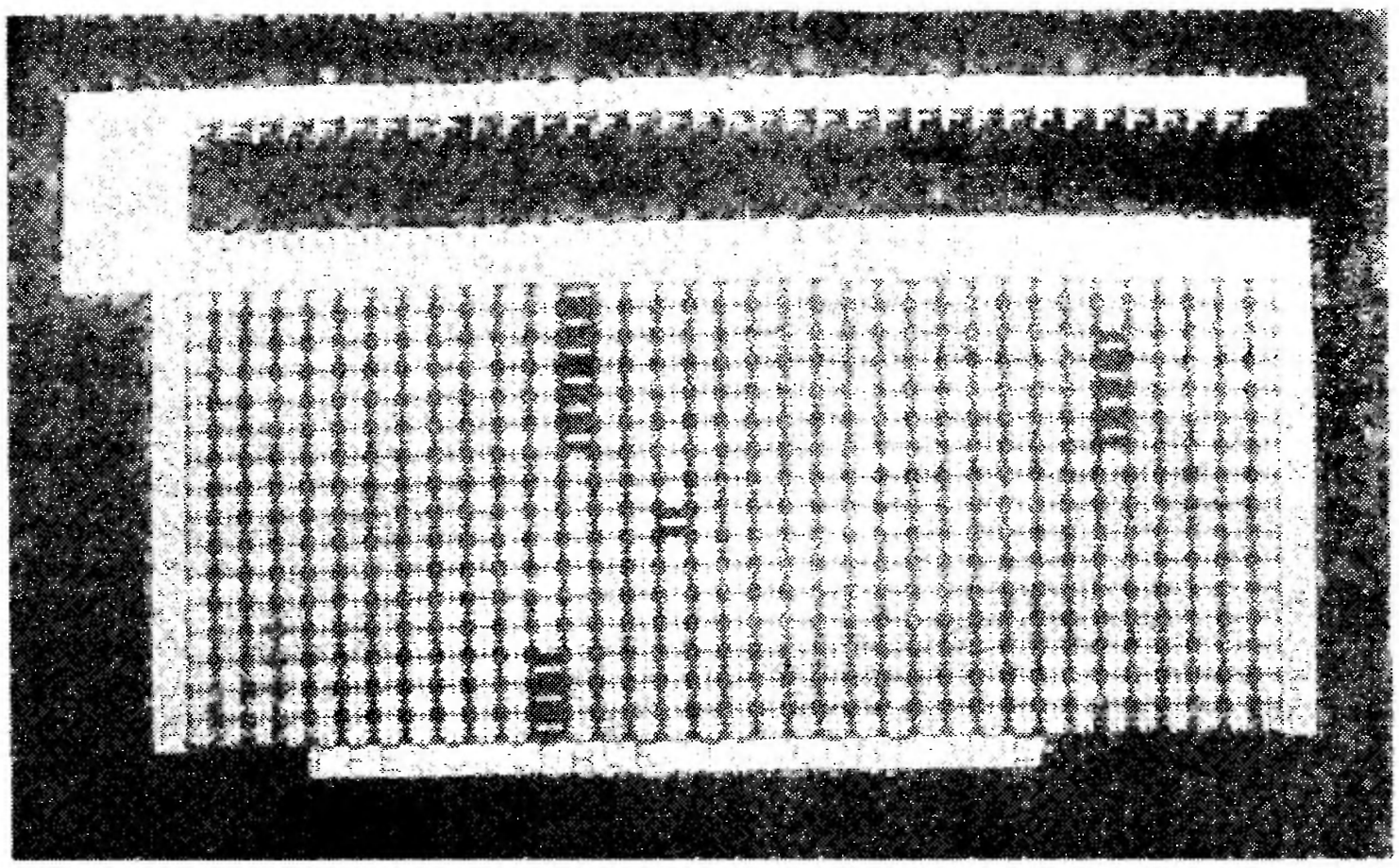
След като функцията се задейства с натискане на клавиша < U >, Locksmith запитва за номера на дискетното устройство и името на изтрития файл. Ако файлът не съществува в справочника, системата съобщава за грешка. Ако върху изтрития файл не е записана нова информация, той се възстановява. При разрушен файл функцията се прекъсва и с натискане на клавиша за интервал се връща главното меню.

ЗАРЕЖДАНЕ НА ДОС ФАЙЛ В ПАМЕТТА (< L > — LOAD DOS FILE INTO MEMORY)

Тази функция зарежда файл от дискета в паметта с натискане на клавиша < L >. След като се посочи името на устройството и на файла, системата го прочита и разполага от адрес $2000. Списъкът на секторите и на пътечките, в които е разположен файлът,



Фиг. 5



Фиг. 6

се записва от адрес $7F00. Така файлът може лесно да се презапише на дискетата отново.

КАРТА НА ДИСКОВОТО ПРОСТРАНСТВО (< M > — SHOW DISK SPACE MAP)

С натискане на клавиша < M > картата на дисковото пространство се изписва на екрана. Заетите сектори са означени с „#“, а свободните с „—“.

ПРОВЕРКА НА ЦЕЛОСТТА НА ТОМА (< V > — VERIFY VTOC INTEGRITY)

Тази функция се задейства с натискане на клавиша < V >, като се проверява списъкът от пътечки и сектори, на които е разположен всеки файл поотделно. Ако дискетата е в ред, на екрана се изобразява карта, подобна на картата на дисковото пространство. При наличие на грешни сектори те се означават по следния начин:

— инверсно „A“, ако секторът е присъединен към файл, но не се използва;

— инверсно „U“, ако секторът е зает, но не е определен към нито един файл. В този случай е необходимо файловете веднага да се копират на празна дискета;

— инверсно „2“, „3“ и т. н. в зависимост от това, към колко файла е присъединен секторът. Файловете трябва също да се копират на празна дискета.

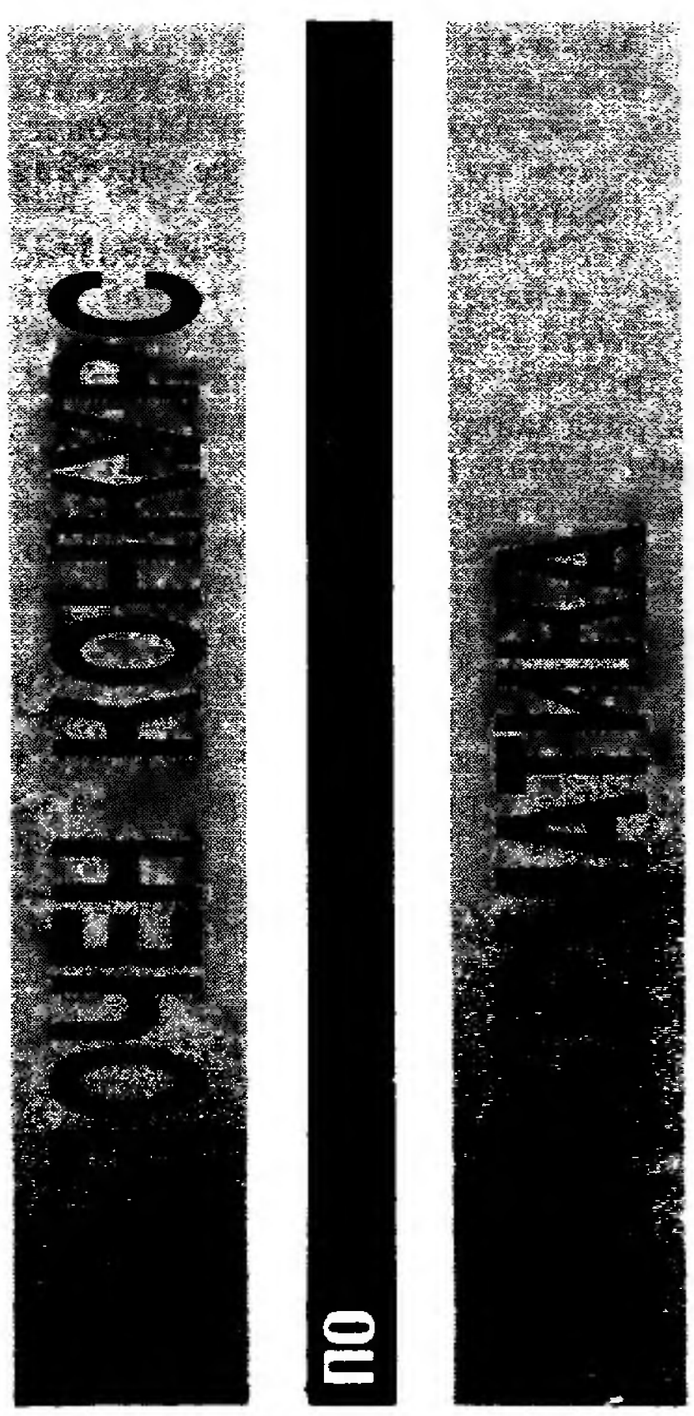
ПРЕМАХВАНЕ НА ДОС ОТ ДИСКЕТА (< R > — REMOVE DOS FROM DISK)

С натискане на клавиша < R > се освобождават пътечки 1 и 2, заемани от ДОС, за запис на друга информация.

ПОДРЕЖДАНЕ НА СПРАВОЧНИКА ПО АЗБУЧЕН РЕД (< A > — ALPHABETIZE CATALOG)

С натискане на клавиша < A > справочникът се прочита от дискетата, подрежда се по азбучен ред и с натискане на клавиша за интервал се записва обратно върху дискетата.

□



Дадени са n различни знака: $a_1, a_2, \ldots a_n$. Построяват се следните n на брой думи: $P_1 = a_1$, $P_2 = P_1 a_2 P_1$, ..., $P_n = P_{n-1} a_n P_{n-1}$ (т. е. $P_1 = a_1$, $P_2 = a_1 a_2 a_1$, $P_3 = a_1 a_2 a_1 a_3 a_1 a_2 a_1$ и т. н.). Да се състави програма, която при зададени номера $i (1 \leqslant i \leqslant n)$ и $k (1 \leqslant k \leqslant n)$ да намира кой знак стои в i-тата позиция на думата P_k.

ЕМИЛ КЕЛЕВЕДЖИЕВ

В една редица са наредени два вида картончета: бели и червени. На всяко бяло картонче е написано по едно цяло число, а на всяко червено — буквата A или буквата B. Дадена е една празна кутия, в която картончетата могат да се поставят само едно върху друго. Едно дете си играе, като взема последователно картончетата от редицата и прави следното.

Ако взетото картонче е бяло, то се поставя в кутията. Ако в кутията има вече други поставени картончета, новото картонче се поставя най-отгоре.

Ако взетото картонче е червено и е с написана буква *A*, тогава, ако кутията е празна или най-горното картонче в кутията е червено, играта завършва аварийно. В противен случай, ако картончето е бяло, числото върху него се изтрива и се написва число, с единица по-малко. Взетото червено картонче не се слага в кутията, а се изхвърля.

Ако взетото картонче е червено и е с написана буква *B*, тогава, ако в кутията няма поне две картончета или най-горните две картончета не са и двете бели, играта завършва аварийно. В противен случай числото върху най-горното картонче се изтрива и се написва число, равно на сумата на изтритото число и числото от по-долното картонче. По-долното картонче се изхвърля. Взетото червено картонче също се изхвърля.

Когато редицата от картончета се изчерпи и в кутията има само едно бяло картонче, казваме, че играта завършва нормално. В противен случай играта завършва аварийно.

Съставете програма, която за всяка дадена редица от картончета да дава съобщения:

1. Дали играта ще завърши нормално, или аварийно.
2. Какво е написано върху картончето, което е най-отгоре в кутията при нормално или аварийно завършване на играта.

ЕМИЛ КЕЛЕВЕДЖИЕВ

Решението трябва да съдържа текста на програмата, обосновка на алгоритъма и описание на начина на работа с нея. Ограничение на езика за програмиране няма. На тези, които се колебаят, прероръчваме Паскал. Желателно е програмата да е записана на дискета за микрокомпютър Правец – 16, Правец – 82 или съвместимите с тях. При оценяване на решенията освен правилността им ще се вземат предвид тяхната ефективност и евентуалните идеи за обобщения.

Решението на всяка задача трябва да се изпрати в отделен плик, като се посочат пълни данни за участника – трите имена, училище (ВУЗ), клас (курс) и точен адрес за кореспондиране (по възможност и телефонен номер). За да бъде върната по пощата, дискетата трябва да е подходящо опакована и на отделен плик да е написан само вашият адрес.

Срокът за изпращане на решенията е два месеца след излизането на списанието. Адресът е:

1133 София
ул. „Акад. Г. Бончев“, бл. 8
Единен център по математика и механика КС за ТНТМ, за конкурса по информатика

За всяка задача се публикува отделно класиране на участниците по точки и се раздават следните награди.

Първа награда – 50 лева
Втора награда – 30 лева
Трета награда – 20 лева

Ще има и годишно класиране със специални награди, които ще бъдат обявени допълнително.

Най-добрите решения на участниците ще бъдат публикувани в списанието.

ЗАДАЧА 6 ОТ 1988 Г.

КЛАСИРАНЕ

1. Пламен Петров от София – 10 т. (50 лв.)
2. Деян Савов от Кърджали – 9 т. (25 лв.)
3. Лъчезар Христов от София – 9 т. (25 лв.)
4. Емил Джалев от Благоевград – 6 т.
5. Александър Лефтеров от София – 4 т.
6. Павел Атанасов от Варна – 4 т.
7. Венелин Спасов от Варна – 2 т.

Монетната система на една държава се състои от n различни по вид монети със стойности $s_1, s_2, \ldots, s_n$ единици. Дадена е парична сума s единици $(0 < s < 100)$.

А) Напишете програма, която пресмята по колко различни начина дадената сума може да бъде размената с монети, ако се допуска:

1) да се използва всеки вид монета не повече от веднъж;

2) да няма ограничение на броя на използваните монети:

Б) Напишете програми, които генерират всички възможни начини за размяняне според двете подточки на т. А.

За информация даваме сведения за монетните системи на:
НРБ: $n = 6$ и монетите са от 1, 2, 5, 10, 20, 50 единици;
СССР: $n = 8$ и съответно 1, 2, 3, 5, 10, 15, 20, 50;
САЩ: $n = 5$ и съответно 1, 5, 10, 25, 50.

```
program MONEY(input,output); { PROBLEM 6 }
const MAX=100;
var COUNT,S:array[0..MAX] of integer;
    I,J,K,N,S1,SUM:integer;

procedure A(COND:integer);
var I,J,K,L,STEP:integer;
begin
  for I:=0 to S1 do COUNT[I]:=0;
  J:=1;
  while ((S1>=S[J])and(J<=N)) do
    begin
      K:=(2-COND)*(S1-S[J])+COND-1;
      STEP:=(2*COND)-3;
      repeat
        COUNT[K+S[J]]:=COUNT[K+S[J]]+COUNT[K];
        K:=K+STEP;
      until((K<1)or(K>(S1-S[J])));
      L:=S[J];
      repeat
        COUNT[L]:=COUNT[L]+1;
        L:=L+S[J];
      until(L>S1)or(COND=1);
      J:=J+1;
    end;
  WRITELN('ТЪРСЕНИЯТ БРОЙ НАЧИНИ В т.А) подт.',COND,') Е:',COUNT[S1]);
end;
```

```
procedure B(COND:integer);
var FIRST:boolean;
    I,J,K,SUM,NUM:integer;
begin
 writeln('КОМБИНАЦИИ ЗА т.В) подт.',COND,') :');
 for I:=1 to N do COUNT[I]:=0;
 J:=N; SUM:=0; NUM:=0;
 repeat
  COUNT[J]:=(S1-SUM) div S[J];
  if (COND=1) and (COUNT[J]>1) then COUNT[J]:=1;
  SUM:=SUM+COUNT[J]*S[J];
  if SUM=S1 then
   begin
    NUM:=NUM+1; FIRST:=true;
    write(NUM,'. ');
    for K:=1 to N do
     if COUNT[K]>0 then
     begin
      if FIRST=true then FIRST:=false
      else write(' + ');
      write(COUNT[K],'x',S[K]);
     end;
     if (NUM mod 24)=0 then readln;
     writeln;
    COUNT[J]:=COUNT[J]-1; SUM:=SUM-S[J];
   end;
  J:=J-1;
 while J=0 do
  begin
   repeat
    J:=J+1;
   until (COUNT[J]>0) or (J=N);
   COUNT[J]:=COUNT[J]-1; SUM:=SUM-S[J]; J:=J-1;
  end;
 until SUM<0;
end;

begin { MAIN }
 repeat
   WRITE('БРОЙ МОНЕТИ N=');
   READLN(N);
 until((N>0)and(n<MAX));
 WRITELN('МОЛЯ ВЪВЕДЕТЕ МОНЕТИТЕ В НАРАСТВАЩ РЕД !');
 S[0]:=0;
 for I:=1 to N do
   repeat
     WRITE('S[',I,']=');
     READLN(S[I]);
   until((S[I]>0)and(S[I]>S[I-1]));
 repeat
   WRITE('ПАРИЧНА СУМА S=');
   READLN(S1);
 until((S1>0)and(S1<MAX));
 A(1);
 A(2);
 B(1);
 B(2);
end.
```

РЕШЕНИЕ

Интерес представляват различните подходи, които са използвали участниците в конкурса при решаване на условие А.

Деян Савов е разгледал задачата като диофантово уравнение $a_1x_1 + a_2x_2 + \ldots + a_nx_n = c$, където c е паричната сума, $a_1, a_2, \ldots, a_n$ са стойностите на монетите и $x_1, x_2, \ldots x_n$, са бройките на монетите от всеки вид. По подходящ начин това уравнение е сведено до уравнението $ax + by = 1$, в което a и b са взаимно прости. Всички решения на последното уравнение се дават от формулите: $x = x_0 + bt$ и $y = y_0 - at$, където x_0 и y_0 е едно кое да е решение, а t пробягва всички цели числа. Авторът е използвал ограниченията, които се налагат от условието на задачата за x и y, за да определи кои стойности може да пробяга t. Броят на тези стойности дава решението на условие А.

Лъчезар Христов е използвал свойствата на функцията $f(v,k)$, чиято стойност е равна на броя начини, по които паричната сума v може да бъде разменена чрез монетите $s_1, s_2, \ldots, s_k$. Според условие А, подточка 1 сумата v може да бъде разменена или като се използва една монета s_k, или само с монетите $s_1, s_2, \ldots s_{k-1}$. Следователно $f(v,k) = f(v - s_k, k-1) + f(v, k-1)$. В предложената от него програма е построено рекурсивно пресмятане на f, като е използвано още, че $f(0,k) = 1$, $f(m,k) = 0$ при $m < 0$ и $f(v,1)$ има стойност 1 при $v = s$ и стойност 0 в противен случай. При решаването на подточка 2 от условие А е постъпено аналогично, като рекурсивната формула е $f(v,k) = f(v - s_k, k) + f(v, k-1)$

Победителят Пламен Петров е предложил оригинално решение на задачата. По-долу публикуваме неговата програма без никакви изменения. Процедурите *A* и *B* решават съответно условията А и Б на задачата, като параметърът в скобите им е *1* или *2* в зависимост от подточките на условията. Процедурата *A* е достатъчно кратка, за да я коментираме, а за процедурата *B* можем да отбележим, че реализира идеята за *n*-те вложени цикъла. Програмата е написана на Турбо Паскал за Правец – 16.

ЕМИЛ КЕЛЕВЕДЖИЕВ

Б. Р.: Публикуваме решението на шеста задача, защото решението на пета задача е много по-обемисто и тъй като за него няма място в броя, ще го поместим в следващия брой.

ЧИСЛЕНО РЕШАВАНЕ НА КУБИЧНО УРАВНЕНИЕ

АТАНАС АВРАМОВ
СТЕФАН ГРОЗЕВ

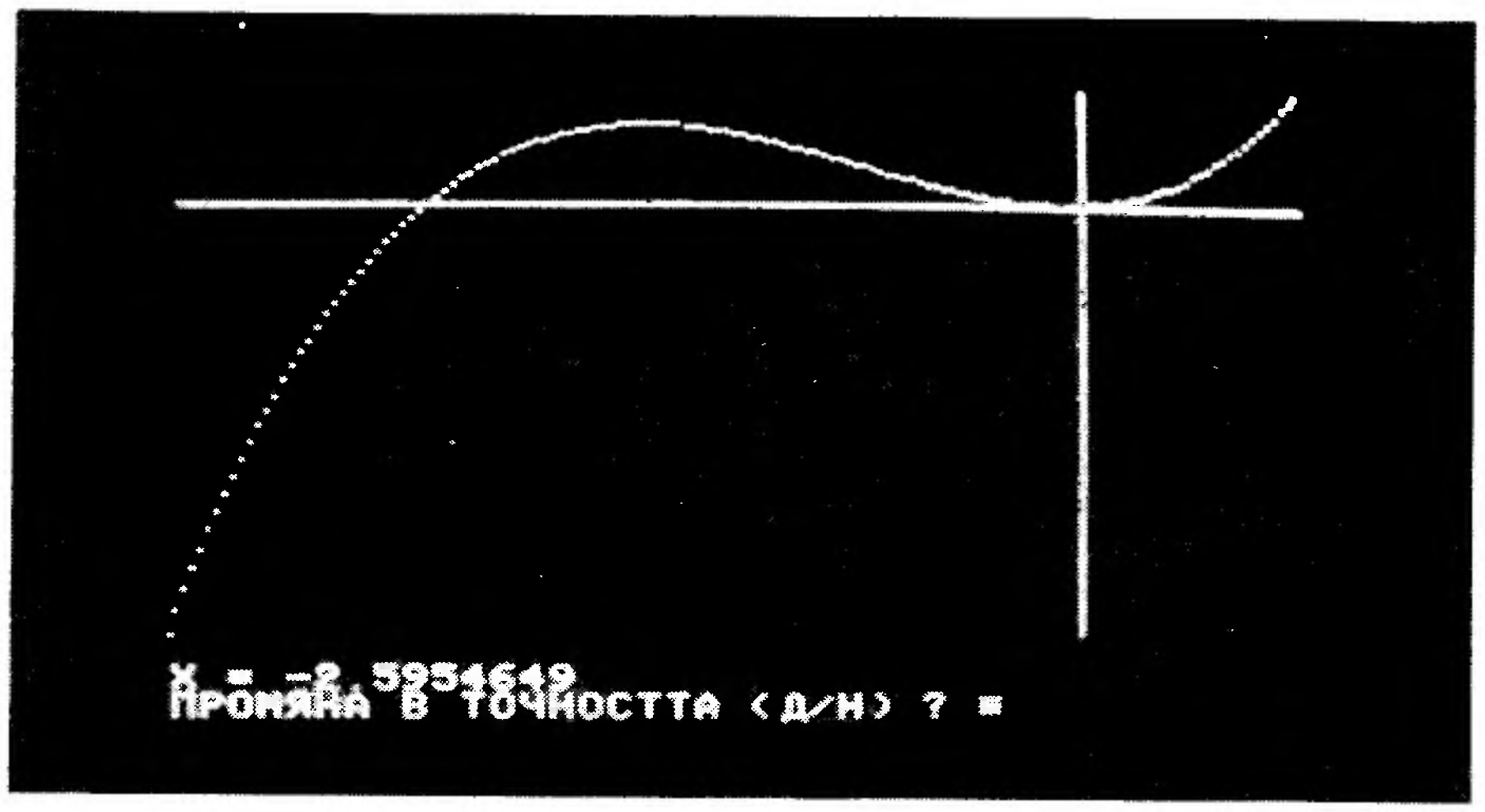
Една от най-често срещаните приложни математически задачи е решаване на уравнения от вида

(1) $$f(x)=0$$

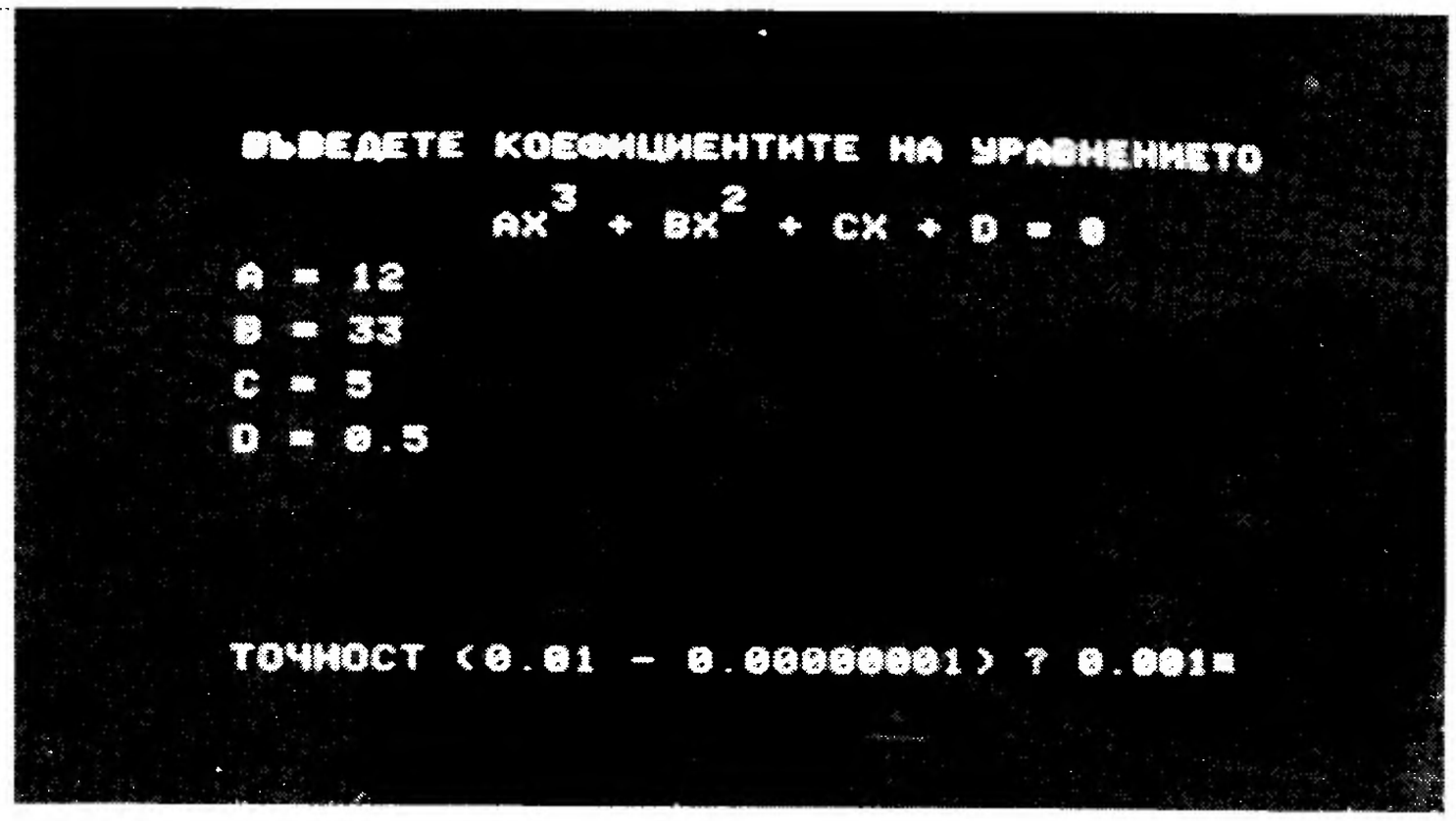
Ако $f(x)$ е полином на x, уравнението се нарича алгебрично. В средния курс по математика се изучават подробно алгебрични уравнения от I и II степен, които се наричат съответно линейни и квадратни уравнения. На базата на квадратните уравнения е разработена и темата за биквадратните уравнения, които са специален клас алгебрични уравнения от IV степен. Далеч от вниманието на учениците обаче остават алгебричните уравнения от III степен или т. нар. кубични уравнения. Тук се разглежда числено решаване на кубични уравнения с реални коефициенти и възможностите за реализиране на съответния алгоритъм на Бейсик за Правец – 82.

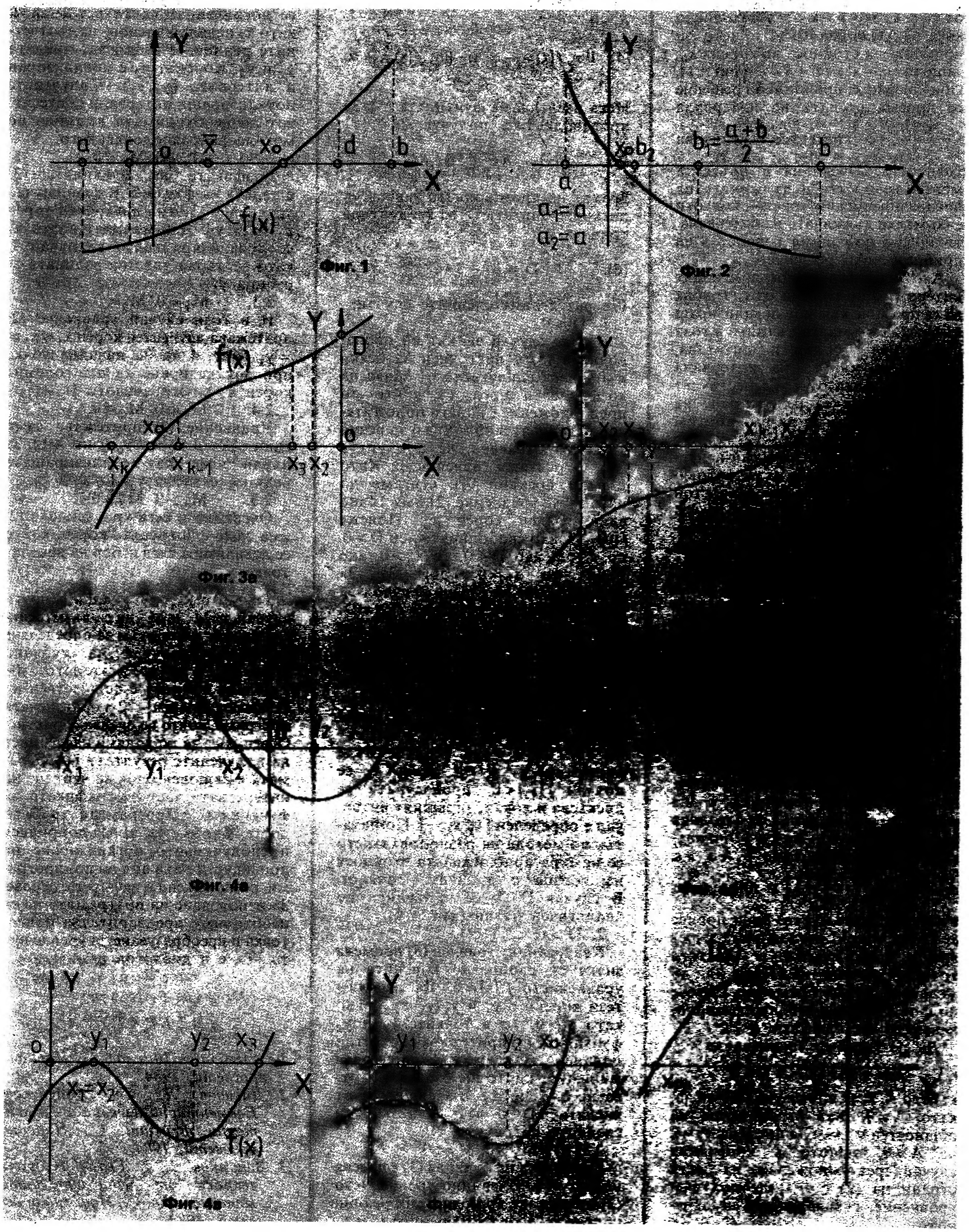
Под решаване на уравнение се разбира намирането на корена (корените) му чрез неговите коефициенти. На всеки ученик са добре известни формулите за решаване на линейно и квадратно уравнение. Подобни формули, познати като формули на Кардано, съществуват и за кубичното уравнение. Но тяхното използване води до сложни пресмятания, които ги правят неподходящи за работа [2]. В такива случаи се търсят методи за приближено решаване на уравнението с предварително зададена точност. Последното твърдение означава, че ако x_0 е точната стойност на търсения корен на дадено уравнение и ε е желаната абсолютна точност (обикновено достатъчно малко положително число, отразяващо изискванията на практиката), то всяко число $\bar{x}$, за което е изпълнено $|\bar{x}-x_0|<\varepsilon$, е приближение с желаната точност на корена x_0. Оказва се, че кубичните уравнения с реални коефициенти могат да бъдат решени със знанията, получени в училищния курс по алгебра.

Ще припомним теоремата на Болцано [1]: ако функцията $f(x)$ е непрекъсната в интервала [a, b] и в краищата му приема стойности с различни знакове, то уравнението (1) притежава поне един реален корен в този интервал. Недостатък на теоремата е, че тя не определя броя на корените на съответното уравнение, нито посочва начин, чрез който те могат да бъдат намерени. Очевидно е едно следствие от тази теорема: ако функцията $f(x)$ е непрекъсната и монотонна (растяща или намаляваща) в интервала [a, b], то в този интервал уравнение (1) има точно един корен. Определянето на интервал, в който уравнението има точно един корен, се нарича отделяне (локализиране) на корена, а интервалът – локализиращ. Явно е, че ако интервалът [a, b] е локализиращ и интервал $[c,d]<[a,b]$ и $f(c).f(d)<0$, то интервалът [c, d] е също локализиращ за дадения корен (фиг. 1).



Първата задача при числения решаване на уравнение (1) е да се установи броят на корените му и те да се локализират. Тази задача може да бъде решена, като се определят интервалите от дефиниционната област на функцията, в които тя е монотонна. Втората задача се състои в уточняването на корените до желаната точност ε. Нека коренът x_0 е локализиран в интервал [a, b]. Тогава, ако дължината на интервала е по-малка





Фиг. 1

Фиг. 2

Фиг. 3а

Фиг. 4а

Фиг. 4в

от ε, т. е. ако $b-a<\varepsilon$, всяко число $\bar{x}\in[a,b]$ може да бъде прието за приближена стойност на корена x_0, защото $|\bar{x}-x_0|<b-a<\varepsilon$ (фиг. 1). Обикновено за приближено решение се приема средата на интервала $\bar{x}=\frac{a+b}{2}$, с което се увеличава точността. Очевидна е връзката между точността и дължината на локализиращия интервал – колкото е по-малка дължината на интервала, толкова е по-голяма точността на приближеното решение.

Ако дължината на локализиращия интервал е по-голяма от ε, трябва да се премине към нов локализиращ интервал с по-малка дължина и този процес да продължи дотогава, докато дължината на интервала стане по-малка от ε. Съществуват много методи за реализиране на тази процедура. За да разработим програма за Правец – 82, ще се спрем на най-простия и най-непретенциозния – метода на разполовяването. Същността му се състои в следното: нека коренът x_0 на уравнение (1) е локализиран в интервал $[a,b]$, като $b-a>\varepsilon$. Изчисляваме $f\left(\frac{a+b}{2}\right)$. Ако $f\left(\frac{a+b}{2}\right)=0$, то $x_0=\frac{a+b}{2}$ и коренът е намерен точно. В противен случай точно един от двата интервала $\left[a,\frac{a+b}{2}\right]$ и $\left[\frac{a+b}{2},b\right]$ ще бъде локализиращ за корена x_0. По този начин построихме интервал $[a_1,b_1]$ (черт. 2), локализиращ корена x_0, с два пъти по-малка дължина от изходния интервал. Ако $b_1-a_1<\varepsilon$, уточняването е завършено. Ако $b_1-a_1>\varepsilon$, намираме $f\left(\frac{a_1+b_1}{2}\right)$ и продължаваме по същия начин. Този процес е краен – при него винаги чрез краен брой стъпки се достига точното значение на корена или до желаното му приближение. Повече подробности за метода на разполовяването ще намерите в 3.

Да разгледаме уравнението

(2) $\quad Ax^3+Bx^2+Cx+D=0,$

където A, B, C и D са реални числа, като $A\neq 0$. Без ограничение на общността можем да предполагаме, че $A>0$, защото в противния случай чрез умножаване на двете страни на (2) с -1 ще получим уравнение, еквивалентно на даденото.

Да означим

(3) $\quad f(x)=Ax^3+Bx^2+Cx+D.$

От средния курс по математика е известно, че функцията (3) е непрекъсната за всяко x. Понеже $A>0$,

(4) $\lim\limits_{x\to-\infty} f(x)=-\infty$ и $\lim\limits_{x\to+\infty} f(x)=+\infty$

Нека изследваме функцията (3). За производната ѝ получаваме

(5) $\quad f'(x)=3Ax^2+2Bx+C.$

Знакът на производната зависи от дискриминантата D′ на квадратния тричлен (5):

(6) $\quad D'=B^2-3AC$

Ще разгледаме следните случаи:

1. $D'\leqslant 0$.

В този случай знакът на производната (5) е постоянен за всяко x и зависи единствено от знака на коефициента пред x^2, който по предположение е винаги положителен. Следователно $f'(x)\geqslant 0$ за всяко x, което означава, че функцията $f(x)$ е растяща навсякъде. Като вземем предвид (4), стигаме до извода, че уравнение (2) има точно един реален корен x_0. Понеже $f(0)=D$, ако свободният член $D>0$, то $x_0<0$; ако $D<0$, то $x_0>0$, ако $D=0$, то $x_0=0$ (фиг. 3). За локализиране на корена се използва функционалната стойност $f(0)=D$. Например при $D>0$ (фиг. 3а) тръгваме от точката 0, движим се наляво и построяваме редица $x_1=0$, $x_2=-1$, $x_3=-2,\ldots$. За всеки член на тази редица проверяваме условието $f(x_n)\geqslant 0$. Ако за някое x_n $f(x_n)=0$, то $x_n=x_0$, т. е. x_n е коренът на (2). Щом това условие бъде нарушено, т. е. за някое x_k се получи $f(x_k)<0$, процедурата се прекъсва и локализиращият интервал е определен $[x_k,x_{k-1}]$. По-нататък по метода на разполовяването се намира приближената стойност на корена с желаната точност. В случая $D<0$ се процедира по аналогичен начин (фиг. 3 б).

2. $D'>0$.

Квадратният тричлен (5) променя знака си съобразно корените на уравнението $f'(x)=0$. Да означим тези корени с y_1 и y_2 $(y_1<y_2)$. Тъй като $A>0$, то в точките y_1 и y_2 функцията (3) притежава съответно локален максимум и локален минимум, които ще означим съответно с M_1 и M_2. В зависимост от числата M_1 и M_2 са възможни следните подслучаи:

2.1. $\quad M_1>0,\ M_2<0.$

Графиката на функция (3) е дадена на фиг. 4 а. Очевидно уравнението (2) притежава три различни реални корена x_1, x_2 и x_3. Корените x_1 и x_3 се локализират както в случай 1, като за начални членове на съответните редици се използват числата y_1 и y_2. Коренът x_2 е локализиран в интервала $[y_1,y_2]$. Получените локализиращи интервали се стесняват до необходимата дължина по метода на разполовяването.

2.2. $\quad M_1>0,\ M_2=0$

Графиката на функцията (3) е дадена на фиг. 4 б. В този случай уравнение (2) има двукратен корен $x_2=x_3=y_2$. Най-малкият корен x_1 се локализира както в случай 1, като за първи член на редицата се избира y_1.

2.3. $\quad M_1=0,\ M_2<0.$

И в този случай уравнение (2) притежава двукратен корен $x_1=x_2=y_1$ (фиг. 4 в). За начална точка при локализиране на най-големия корен x_3 служи числото y_2.

2.4. $\quad M_1<0,\ M_2<0.$

Уравнение (2) притежава само един реален корен x_0 (фиг. 4 г). За първи член на локализиращата редица се взема y_2.

2.5. $\quad M_1>0,\ M_2>0.$

Очевидно и сега уравнението (2) има само един реален корен, който се локализира след избор на начална точка y_1.

Тъй като $M_1>M_2$ (защо?), други случаи за M_1 и M_2 не са възможни. Описаният алгоритъм за определяне броя на корените на кубично уравнение и пресмятането им с желана точност е осъществен на Бейсик за Правец – 82 в приведената програма, която го следва стриктно. Освен това за по-голяма нагледност на получените резултати в програмата е включен блок за чертане на интересната част от графиката на функцията. Програмата работи доста бързо. За това допринася и сполучливото организиране на процедурите във вид на подпрограми, разположени в началните редове. За използване на програмата не са необходими предварителна подготовка и преобразуване на уравнението. Тя е в диалогов режим.

ЛИТЕРАТУРА

1. *Алгебра за 10. клас. С., Народна просвета, 1984.*
2. *Давидов, Л., Ст. Додунеков. Елементарна алгебра и елементарни функции. С., Народна просвета, 1984.*
3. *Запрянов, З., Г. Геров. Избрани въпроси от изчислителната математика. С., Народна просвета, 1973.*

```
10  REM  ВЪВЕЖДАНЕ
20  HOME : PRINT "ВЪВЕДЕТЕ КОЕФИ
    ЦИЕНТИТЕ НА УРАВНЕНИЕТО"
30  VTAB 3: HTAB 12: PRINT "3
      2"
40  HTAB 10: PRINT "AX  + BX  +
    CX + D = 0"
50  PRINT : INPUT "A = ";A(1): PRINT
    : INPUT "B = ";A(2): PRINT :
     INPUT "C = ";A(3): PRINT : INPUT
    "D = ";A(4)
60  IF A(1) <  > 0 THEN 90
70  PRINT : PRINT : PRINT "НЕКОР
    ЕКТНА ИНФОРМАЦИЯ - A = 0 !"
    : FOR J = 1 TO 2000: NEXT : GOTO
    20
80  REM   ДЕФИНИРАНЕ НА ФУНКЦИЯТ
    А
90 Q = A(1)
100  IF A(1) > 0 THEN 120
110  FOR I = 1 TO 4:A(I) =  - A(
    I): NEXT I
120  DEF  FN F(X) = A(1) * X * X
     * X + A(2) * X * X + A(3) *
    X + A(4)
130  VTAB 20: HTAB 1: CALL  - 86
    8: VTAB 20: HTAB 1: INPUT "Т
    ОЧНОСТ (0.01 - 0.00000001) ?
     ";E
140  IF E > .01 OR E < .00000001
     THEN  PRINT  CHR$ (7);: GOTO
    130
150  GOTO 630
160  REM   ИЗЧИСЛЯВАНЕ НА ЛОКАЛН
    ЗИРАН КОРЕН
170  IF X2 - X1 < E THEN  RETURN

180  IF ( FN F((X1 + X2) / 2)) *
    ( FN F(X1)) < 0 THEN X2 = (X
    1 + X2) / 2: GOTO 170
190  IF  FN F((X1 + X2) / 2) = 0
     THEN  RETURN
200 X1 = (X1 + X2) / 2: GOTO 170

220  REM  ЛОКАЛИЗИРАНЕ НА НАЙ-МА
    ЛКИЯ КОРЕН.
230 X2 = X2 - 1
240  IF  FN F(X2) > 0 THEN 230
250  IF  FN F(X2) = 0 THEN X1 =
    X2: RETURN
260 X1 = X2:X2 = X2 + 1: GOSUB 1
    70: RETURN
280  REM  ЛОКАЛИЗИРАНЕ НА НАЙ-ГО
    ЛЕМИЯ КОРЕН.
290 X1 = X1 + 1
300  IF  FN F(X1) < 0 THEN 290
310  IF  FN F(X1) = 0 THEN X2 =
    X1: RETURN
320 X2 = X1:X1 = X1 - 1: GOSUB 1
    70: RETURN
340  REM  ЧЕРТАНЕ НА ГРАФИКАТА.
350 A = Z1 - 1:B = Z3 + 1
360  IF A <  = 0 THEN 380
370 A = 0
380  IF B >  = 0 THEN 400
390 B = 0
400  IF  FN F(A) <  = M2 THEN C =
     FN F(A): GOTO 420
410 C = M2
420  IF  FN F(B) >  = M1 THEN P =
     FN F(B): GOTO 460
430 P = M1
440  IF P >  = 0 THEN 460
450 P = 0
460  IF Q > 0 THEN 480
470 L = C:C =  - P:P =  - L
480 M = 279 * A / (A - B):N = 15
    9 * P / (P - C)
490  HOME : HGR : HCOLOR= 3
500  HPLOT 0,N TO 279,N: HPLOT M
    ,159 TO M,0
510  IF Q > 0 THEN 570
520  FOR X = A TO B STEP (B - A)
     / 140
530 Y =  - FN F(X)
540 T = 279 * (X - A) / (B - A):
    M = 159 * (P - Y) / (P - C):
     HPLOT T,M
550  NEXT X: RETURN
570  FOR X = A TO B STEP (B - A)
     / 140
580 Y =  FN F(X)
590 T = 279 * (X - A) / (B - A):
    M = 159 * (P - Y) / (P - C):
     HPLOT T,M
600  NEXT X: RETURN
620  REM  ИЗСЛЕДВАНЕ НА КУБИЧНАТ
    А ФУНКЦИЯ.
630 D = A(2) * A(2) - 3 * A(1) *
    A(3)
640  IF D <  = 0 AND A(4) > 0 THEN
    X2 = 0: GOSUB 230:B = 1:A =
    (X1 + X2) / 2 - 1:C =  FN F(
    A):P =  FN F(B): GOSUB 440: GOTO
    760
650  IF B <  = 0 AND A(4) < 0 THEN
    X1 = 0: GOSUB 290:A =  - 1:B
     = (X1 + X2) / 2 + 1:C =  FN
    F(A):P =  FN F(B): GOSUB 440
    : GOTO 700
660  IF B <  = 0 THEN A =  - 2:B
     = 2:C =  FN F(A):P =  FN F(
    B): GOSUB 440: GOTO 820
670 Y1 = ( - A(2) -  SQR (D)) /
    (3 * A(1)):Y2 = ( - A(2) +  SQR
    (D)) / (3 * A(1))
680 M1 =  FN F(Y1):M2 =  FN F(Y2
    )

690  IF M1 > 0 AND M2 < 0 THEN X
    2 = Y1: GOSUB 230:Z1 = (X1 +
    X2) / 2:X1 = Y1:X2 = Y2: GOSUB
    170:Z2 = (X1 + X2) / 2:X1 =
    Y2: GOSUB 290:Z3 = (X1 + X2)
     / 2: GOSUB 350: GOTO 830
700  IF M1 > 0 AND M2 = 0 THEN X
    2 = Y1: GOSUB 230:Z1 = (X1 +
    X2) / 2:Z2 = Y2:Z3 = Y2: GOSUB
    350: GOTO 830
710  IF M1 = 0 AND M2 < 0 THEN X
    1 = Y2: GOSUB 290:Z1 = Y1:Z2
     = Y1:Z3 = (X1 + X2) / 2: GOSUB
    350: GOTO 830
720  IF M1 < 0 AND M2 < 0 THEN X
    1 = Y2: GOSUB 290:Z3 = (X1 +
    X2) / 2:Z1 = Y1: GOSUB 350: GOTO
    810
730  IF M1 > 0 AND M2 > 0 THEN X
    2 = Y1: GOSUB 230:Z1 = (X1 +
    X2) / 2:Z3 = Y2: GOSUB 350: GOTO
    810
740  REM  ИЗВЕЖДАНЕ НА КОРЕНИТЕ.
750  VTAB 22: PRINT "X = ";X2: GOTO
    840
760  IF  FN F((X1 + X2) / 2) = 0
     THEN 810
770  GOTO 750
780  IF  FN F((X1 + X2) / 2) = 0
     THEN 810
790  GOTO 800
800  VTAB 22: PRINT "X = ";X1: GOTO
    840
810  VTAB 22: PRINT "X = ";(X1 +
    X2) / 2: GOTO 840
820  VTAB 22: PRINT "X = 0": GOTO
    840
830  VTAB 21: PRINT "X1 = ";Z1;"
      ";"X2 = ";Z2: PRINT "X3 =
    ";Z3
840  VTAB 23: HTAB 1: PRINT "ПРО
    МЯНА В ТОЧНОСТТА (Д/Н) ? ";:
     GET D$
850  IF D$ = "Д" THEN  TEXT : HOME
    : GOTO 130
860  IF D$ = "Н" THEN 880
870  PRINT  CHR$ (7);: GOTO 840
880  VTAB 23: HTAB 1: CALL  - 86
    8: VTAB 23: HTAB 1: PRINT "Н
    ОВО УРАВНЕНИЕ (Д/Н) ? ";: GET
    D$
890  IF D$ = "Д" THEN  TEXT : CLEAR
    : GOTO 20
900  IF D$ = "Н" THEN 920
910  PRINT  CHR$ (7);: GOTO 880
920  TEXT : HOME : VTAB 12: HTAB
    10: PRINT "Д О В И Ж Д А Н Е
     !"
```

СТАВА 3 РЕДА.

ОТ # РЕД	ДО # РЕД	КОНТР. СУМА
10	- 30	2F10
60	- 100	2613
110	- 150	3426
160	- 200	3422
220	- 260	2738
280	- 320	2409
340	- 380	1940
390	- 430	182F
440	- 480	1DD7
490	- 530	1978
540	- 590	2619
600	- 650	485E
660	- 700	6286
710	- 750	4C8F
760	- 800	1044
810	- 850	2CBB
860	- 900	2521
910	- 920	0F19

КОНТР. СУМА Е		154C

СОФТУЕР

ИЗЧИСЛЯВАНЕ НА ФИЛТРИ НА ЧЕБИШЕВ

В КВ. 7—8.88 беше публикувана програма за изчисление на филтри с характернстична функция полином на Бътъруърт. С въвеждането на минимални промени същата програма може да се използва за изчисление на филтрн с характеристична функция полиом на Чебишев. Програмните редове, които трябва да се отстранят, добавят или променят, са дадени в листинга.

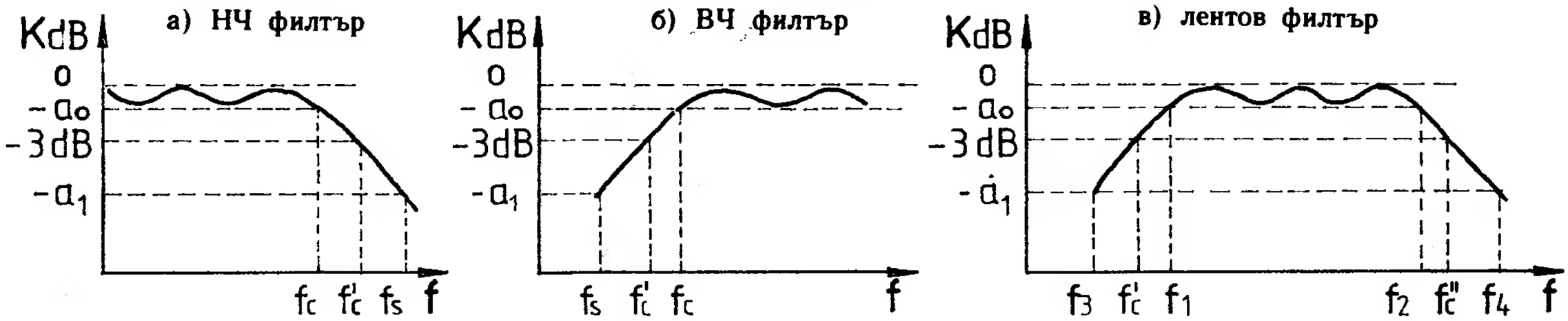
Филтрите на Чебишев имат аналогични схеми с тези на Бътъруърт. Различията между двата типа филтри са в стойностите на елементите и формата на амплитудно-честотните характеристики (АЧХ). Филтрнте на Чебишев имат АЧХ, показани на фиг. 1. В лентата на пропускане характернстиката е равновълнообразна. Броят на навълняванията зависи от порядъка на филтъра. АЧХ на филтрнте на Чебишев имат по-голяма стръмност в преходната област.

При изчисленнето на филтър на Чебишев вместо затихването на честотата на среза се задава исканата неравномерност на АЧХ в лентата на пропускане. Параметрите, които се изчисляват от програмата, са същите както при филтрите на Бътъруърт, като се пресмята допълннтелно и честотата на среза при затихване 3 dB.

Инж. ВЛАДИСЛАВ ГЕОРГИЕВ

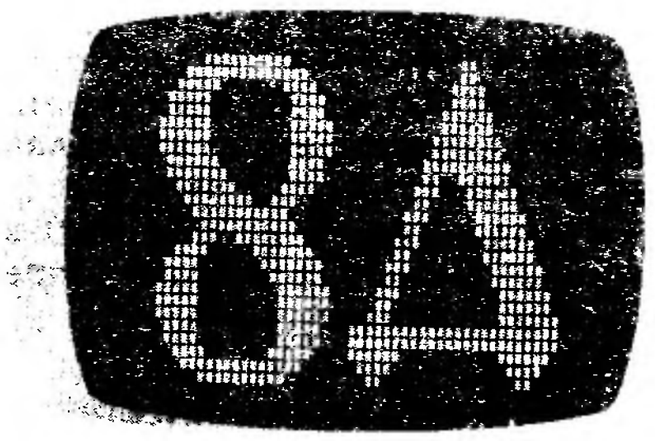
```
2  DEF  FN CH(X) = ( EXP (X) +  EXP
     ( - X)) / 2
3  DEF  FN ACH(X) =  LOG (X +  SQR
     (X ^ 2 - 1))
4  DEF  FN SH(X) = ( EXP (X) -  EXP
     ( - X)) / 2
5  DEF  FN CTH(X) =  SQR (1 +  FN
     SH(X) ^ 2) /  FN SH(X)
6  DEF  FN A(X) =  SIN (PI * (2 *
     X - 1) / (2 * N))
20  PRINT  TAB( 6);"ПРЕСМЯТАНЕ Н
    А ФИЛТЪР НА ЧЕБИШЕВ"
180  PRINT "ВЪВЕДЕТЕ НЕРАВНОМЕРН
    ОСТ НА АЧХ В DB": INPUT A0: IF
    A0 <  = 0 THEN  PRINT "НЕРАВ
    НОМЕРНОСТТА ТРЯБВА ДА Е >0":
     GOTO 180
320 F =  SQR (F / EE)
330  REM ОТСТРАНЯВА СЕ
340 N =  INT ( FN ACH(F) /  FN A
    CH(W0) + 1)
350 A = 10 *  LOG (1 + EE * ( FN
    CH(N *  FN ACH(W0))) ^ 2) /
     LOG (10)
360  IF A0 < 3 THEN K =  FN CH( FN
    ACH( SQR ((10 ^ .3 - 1) / EE
    )) / N)
370  IF C < 5 GOTO 400
380 DK = DF * (K - 1) / 2
390 KA = F0 / DF
450  PRINT  TAB( 10);" ФИЛТЪР НА
     ЧЕБИШЕВ"
510  IF F1 < F2 THEN  PRINT  TAB(
    26);F1: PRINT  TAB( 26);F2: GOTO
    520
515  PRINT  TAB( 26);F2: PRINT  TAB(
    26);F1
520  PRINT "НЕРАВНОМЕРНОСТ НА АЧ
    Х";: PRINT  TAB( 26);A0
534  IF A0 >  = 3 THEN  GOTO 540

535  PRINT "ЧЕСТОТА НА СРЕЗА -3D
    B"; TAB( 26);
536  IF C < 3 THEN  PRINT K * F0
    : GOTO 540
537  IF C < 5 THEN  PRINT F0 / K
    : GOTO 540
538  IF F1 > F2 THEN  PRINT F2 -
    DK: PRINT  TAB( 26);F1 + DK:
     GOTO 540
539  PRINT Fi - DK: PRINT  TAB(
    26);F2 + DK
755 D =  FN SH( LOG ( FN CTH(A0 *
     LOG (10) / 40)) / (2 * N))
760 G1 = 2 *  FN A(1) / D
820 G = G1
870  PRINT  TAB( 24);G / (R1 * N
    ); TAB( 39);"F"
880  GOTO 905
900  PRINT  TAB( 24);G * R1 / N;
     TAB( 39);"H"
905 G1 = 4 *  FN A(I) *  FN A(I +
    1) / ((D ^ 2 + ( SIN (I * PI
     / N)) ^ 2) * G1)
960 G = G1
1000  PRINT  TAB( 24);1 / (KA *
    G * R1 * N); TAB( 39);"F"
1030  GOTO 1075
1050  PRINT  TAB( 24);KA * G / (
    R1 * N); TAB( 39);"F"
1070  PRINT  TAB( 24);R1 / (KA *
    G * N); TAB( 39);"H"
1075 G1 = 4 *  FN A(I) *  FN A(I
     + 1) / ((D ^ 2 + ( SIN (I *
    PI / N)) ^ 2) * G1)
1110  IF P = 1 THEN  PR# 0: END

1140  IF A$ = "Д" OR A$ = "D" THEN
    P = 1: PRINT "ВКЛЮЧЕТЕ ПРИНТ
    ЕРА И НАТИСНЕТЕ КЛАВИШ": GET
    A$: PR# 1: PR# 1: GOTO 410
```

Фиг. 1 АЧХ на филтрите на Чебишев:



КВ 03-04

ЕЛЕКТРОНЕН КАЛЕНДАР — БЕЛЕЖНИК



Инж. ЙОРДАН ЙОРДАНОВ

Всеки делови човек се стреми най-оптимално да разпредели личното си и служебното време, да знае какво е правил на дадена дата или какво ще прави на друга дата. За целта било на бюрото си, било в дамската чанта, или пък в някой от джобовете всеки носи един малък бележник с календар и вписва какво трябва да направи днес, утре или в някой следващ ден.

С навлизането на микрокомпютрите в нашето всекидневие производителите на програмни продукти започнаха да предлагат програми, които да поемат върху себе си рутинната дейност в деловодството и управлението.

Такава програма е и Електронен календар, която се предлага в няколко варианта за микрокомпютри Правец-16 и Правец-82.

Тук предлагам своя разработка на Електронен календар за домашен микрокомпютър Правец-8Д. Тази програма:

— може да съхранява информацията от бележки за всеки ден в течение на една календарна година само с едно зареждане от касетофон (за всяка друга година трябва да се прави ново зареждане, като ограничения за годините до 2000 няма);

— може да формира на екрана на микрокомпютъра месечен календар за кой да е месец от коя да е година до 2000;

— дава възможност за всеки избран ден от годината да се въвеждат до осем събития с дължина до 30 знака;

— дава възможност да се изведе и редактира вече въведеният текст за някои от дните на годината.

С програмата се работи по следния начин.

След като се набере текстът и се стартира с RUN, на екрана се извежда заглавният надпис и компютърът очаква да се въведе текущата дата. Датата се въвежда във формат ден, месец, година. Например, ако датата е 1 януари 1989 година, от клавиатурата трябва да се въведе 010189 и да се натисне RETURN. Програмата прави автоматичен контрол и ако има грешка при въвеждането (например 13-и месец или 33 ден от месеца), отново позиционира курсора в началото и очаква вярната дата. По принцип датата може и да не е точно текущата. Достатъчно е тя да е валидна. Системата впоследствие дава възможност за избор и уточняване на кой да е дата от годината. Добре е обаче датата да е поне от същия месец, за който желаем въвеждане или четене на текст. Това ще спести време за търсене на желания месец.

След това програмата извежда на екрана съобщение и очаква избор, дали първоначално ще се създава масивът с текстовете за дадена година, или е вече създаден и трябва да се зареди от касета.

Ако до този момент не е създадено копие на масива с текстовете на касета, необходимо е да се направи копие на масива за друга година или просто не се изисква работа с текстове, а само извеждане на произволен месечен календар, се отговаря с натискане на клавишите 1 и RETURN.

Ако вече има записан текст за някоя година на касета и от програмата се изисква работа с него, се отговаря с клавиши 2 и RETURN и се изпълнява процедурата по зареждане на масива.

Когато масивът е зареден или е избрано първоначално създаване на масив, програмата извежда месечния календар на текущия месец и позиционира курсора в последния му ред. С помощта на стрелките за движение нагоре или надолу програмата автоматично изчислява и извежда календар за предходен или следващ месец. Така могат да се „разлистват“ месеците от календара за текущата годнна. Когато е изведен месец януари, програмата забранява понататъшно „връщане“ назад. По същия начин, когато е избран декември, не може да се иска преминаване с месец напред.

След като на екрана е изведен желаният месец и искаме да изберем конкретен ден от него, за който да се въведе текст, вместо клавишите — стрелки се натиска RETURN. След последния ден от месеца се извежда показалец (знака „<“), и програмата очаква да й се посочи ден.

Изборът става чрез клавишите — стрелки за движение в четирите посоки. С натискането на някоя стрелка програмата позиционира курсора на нов ден от месеца в посочената посока.

Когато желаният ден е избран от месеца, отново се натиска RETURN. Екранът се изчиства и в горния му ляв ъгъл се изписват датата, месецът и денят от седмицата. След това, ако е посочено първоначално създаване на масива, програмата очаква да се въведе текстът за деня.

Въвеждането на текста става по обикновения начин (компютърът е в режим кирилица). Ако е допусната грешка, редактирането може да се извърши само чрез клавишите и МК — А. За въвеждане на текст са отредени осем реда по 30 знака. Ако текстът от един ред е по-дълъг от 30 знака, програмата автоматично продължава на следващия ред (без натискане на RETURN). Ако текстът е по-къс от 30 знака, за преминаване на нов ред се използува клавишът RETURN. Ако са заети само няколко реда (има и празни), то те „се пропускат“ с последователно натискане на RETURN, без да се въвежда текст.

Когато „се обходят“ всичките осем реда, програмата изчиства екрана и очаква да се продължи с избор за друг ден или да се приключи работата с календара.

При натискане на клавишите 1 и RETURN програмата извежда отново календар за месеца, с който току-що е работено. Изборът на друг месец, ден и т. н. се извършва по описания по-горе начин.

Ако се натиснат клавишите 2 и RETURN, програмата изчиства

екрана и очаква да се постави касетата, където да се запише зареденият в началото или новосъзданият масив.

Ако не е необходим запис на масива върху касета, касетофонът просто не се включва и се изчаква програмата да изчисти екрана и да излезе съобщението „ГОТОВ“.

Програмата работи по следния начин.

От ред 180 до ред 280 се прави проверка за правилността на въведената в началото дата.

От ред 290 до ред 440 се прави изборът на месец н се извежда месечният календар.

От ред 450 до 560 се прави избор на ден от месеца.

От ред 570 до 600 се изчислява избраната дата.

От ред 605 до 640 се извежда текстът за избрания ден (ако има такъв). Текстът за деня се пази в масива G$ (366), като всеки елемент от 255 знака се разглежда като съставен от 8 низа по 30 знака всеки.

От ред 650 до 680 е разположен модулът, управляващ въвеждането на текста за избрания ден.

Подпрограмата от ред 1000 извежда на екрана празния календар за избрания месец.

Подпрограмата от ред 1500 изчислява на кой ден от седмицата отговаря избраната дата.

```
  5 PAPER0:INK7
 10 GRAB:CLS:PRINT CHR$(4):PRINT @5,7;CHR$(27);"ЈЕЛЕКТРОНЕН
КАЛЕНДАР-БЕЛЕЖНИК"
 20 PRINT CHR$(4):PRINT @ 9,11;CHR$(27);"L(C) ИНЖ. И. ИОРДАН
ОВ"
 25 DIM D%(12),G$(366),IM$(12),DJ%(12)
 30 PRINT @ 1,17;"ЗАДАЙТЕ ДНЕШНАТА ДАТА [ДДММГГ]";:INPUT TD$
 40 PRINT CHR$(20)
 50 D%(1) = 31
 60 D%(2) = 28
 70 D%(3) = 31
 80 D%(4) = 30
 90 D%(5) = 31
 100 D%(6) = 30
 110 D%(7) = 31
 120 D%(8) = 31
 130 D%(9) = 30
 140 D%(10) = 31
 150 D%(11) = 30
 160 D%(12) = 31
 170 D$ = "СЪБНЕДПОНВТОСРЯЧЕТПЕТ"
 175 DATA ЯНУАРИ,ФЕВРУАРИ,МАРТ,АПРИЛ,МАЙ,ЮНИ,ЮЛИ,АВГУСТ,СЕПТ
ЕМВРИ,ОКТОМВРИ
 177 DATA НОЕМВРИ,ДЕКЕМВРИ
 180 GR$=RIGHT$(TD$,2):DR$=LEFT$(TD$,2)
 190 MR$=MID$(TD$,3,2):X1 =VAL(MR$+DR$+GR$)
 200 IF X1 <= 0 THEN 30
 210 M = INT(X1/1E4)
 220 IF M > 12 OR M < 1 THEN 30
 230 Y = INT((X1/100 - INT(X1/100))*100+0.5)
 240 IF Y/4 <> INT(Y/4) THEN 260
 250 D%(2) = 29
 260 D2 = INT((X1-(M*1E4+Y))/100)
 270 IF D2 < 1 THEN 30
 280 IF D2 > D%(M) THEN 30
 282 CLS:PRINT "АКО ЗА ПЪРВИ ПЪТ СТАРТИРАТЕ ПРОГРА-"
 283 PRINT "МАТА НАТИСНЕТЕ <1> АКО НЕ - <2>"
 284 INPUT NO:IF NO=2 THEN GOSUB 2000
 285 RESTORE: FOR I=1 TO 12: READ IM$(I):DJ%(I)=SS:SS=SS+D%(
I):NEXT
```

```
 290 GOSUB 1000
 300 YT=6: FOR D2=1 TO D%(M)
 310 GOSUB 1500:IF N=0 THEN XT=27:PRINT @XT,YT;D2;
 320 IF N=1 THEN XT=32:PRINT @XT,YT;D2:IF D2<>D%(M) THEN YT=
YT+4
 330 IF N=2 THEN XT=2:PRINT @XT,YT;D2;
 340 IF N=3 THEN XT=7:PRINT @XT,YT;D2;
 350 IF N=4 THEN XT=12:PRINT @XT,YT;D2;
 360 IF N=5 THEN XT=17:PRINT @XT,YT;D2;
                       |
 370 IF N=6 THEN XT=22:PRINT @XT,YT;D2;
 380 NEXT
 385 K$=KEY$
 390 IF K$=CHR$(10) AND M<12 THEN M=M+1:GOTO 290
 400 IF K$=CHR$(10) AND M=12 THEN 430
 410 IF K$=CHR$(11) AND M>1 THEN M=M-1:GOTO 290
 420 IF K$=CHR$(11) AND M=1 THEN 430
 430 IF K$=CHR$(13) THEN 450
 440 GOTO 385
 450 XT=XT+3:PRINT @XT,YT;"<";
 460 K$=KEY$
 465 XK=XT:YK=YT
 470 IF K$=CHR$(8) AND XT>8 THEN XT=XT-5:PRINT @XK,YK;" ":PR
INT @XT,YT;"<";
 480 IF K$=CHR$(8) AND XT<8 THEN 560
 490 IF K$=CHR$(9) AND XT<32THEN XT=XT+5:PRINT @XK,YK;" ":PR
INT @XT,YT;"<";
 500 IF K$=CHR$(9) AND XT=32THEN 560
 510 IF K$=CHR$(10) AND YT<24 THEN YT=YT+4:PRINT @XK,YK;" ":
PRINT @XT,YT;"<";
 520 IF K$=CHR$(10) AND YT>24 THEN 560
 530 IF K$=CHR$(11) AND YT> 8 THEN YT=YT-4:PRINT @XK,YK;" ";
:PRINT @XT,YT;"<";
 540 IF K$=CHR$(11) AND YT< 8 THEN 560
 550 IF K$=CHR$(13) THEN 570
 560 GOTO 460
 570 IF SCRN(XT-1,YT)=32 THEN D2=SCRN(XT-2,YT) AND 15:GOTO 6
00
 580 D2=(SCRN(XT-2,YT) AND 15)*10+(SCRN(XT-1,YT) AND 15)
 600 GOSUB 1500:CLS:PRINT D2;" ";IM$(M);" - ";A$
 605 PRINT:PRINT:P=1:Q1$=""
 610 ET=DJ%(M)+D2:IF G$(ET)="" THEN 650
 620 FOR K=1 TO 8:FOR L=1 TO 30
 630 PRINT MID$(G$(ET),P,1);:P=P+1:NEXT
 640 PRINT:PRINT:NEXT:P=1
 650 PRINT @2,3;:FOR K=1 TO 8:FOR L=1 TO 30
 660 GET Q$:IF Q$<>CHR$(13) THEN 665
 662 FOR L=LTO30:Q1$=Q1$+" ":P=P+1:PRINT " ";:NEXT:GOTO 670
 665 IF Q$=CHR$(1) THEN Q$=MID$(G$(ET),P,1)
 668 P=P+1:PRINT Q$;:Q1$=Q1$+Q$:NEXT
 670 PRINT:PRINT:NEXT
```

```
680 G$(ET)=Q1$
690 CLS:PRINT "НАТИСНЕТЕ КЛАВИШ <1> ЗА ДРУГ ДЕН ИЛИ"
700 PRINT "КЛАВИШ <2> ЗА КРАЙ"
710 INPUT KO:IF KO = 1 THEN 290
720 IF KO=2 THEN 750
730 GOTO 690
750 CLS:PRINT "ПРИГОТВЕТЕ КАСЕТКАТА ЗА ЗАПИС НА"
760 PRINT "МАСИВА С ТЕКСТА И НАТИСНЕТЕ <RETURN>"
770 STORE G$,"1988Г.":CLS:RELEASE:END
1000 CLS:LI=LEN(IM$(M)):MP=20-INT(LI/2):PRINT @MP,1;IM$(M)
1010 FOR I = 4 TO 24 STEP 4: FOR J=1 TO 36: PRINT @J,I;"*":
NEXT:NEXT
1020 FOR I=4 TO 24: FOR J=1 TO 39 STEP5: PRINT @J,I;"*":NEX
T:NEXT
1030 PRINT @1,3;" ПОН  ВТО  СРЯ  ЧЕТ  ПЕТ  СЪБ  НЕД":RETUR
N
1500 AM=M:AY=Y:A =0 :IF Y>1900 THEN 1520
1510 Y=Y+1900
1520 IF M>2 THEN 1550
1530 M=M+12
1540 Y=Y-1
1550 A=D2+2*M+INT(0.6*(M+1))+Y+INT(Y/4)
1560 N=A-INT(Y/100)+INT(Y/400)+2
1570 N=INT((N/7-INT(N/7))*7+0.5)
1580 A$=MID$(D$,(N*3)+1,3):M=AM:Y=AY:RETURN
2000 PRINT "ПРИГОТВЕТЕ КАСЕТКАТА С МАСИВА С "
2010 PRINT "ТЕКСТА И НАТИСНЕТЕ <RETURN>"
2020 GET V$:RECALL G$,"":RETURN
```

ИГРИ



СЛАЛОМ

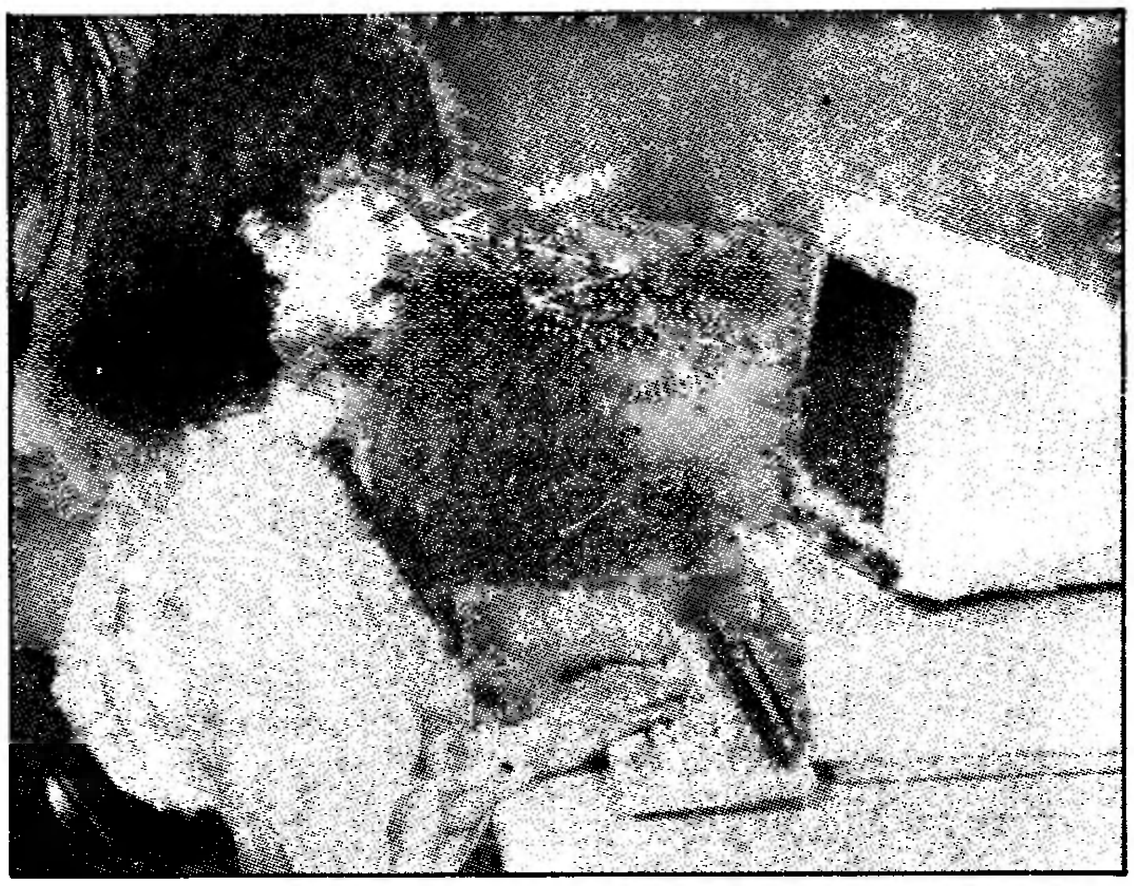
Играта СЛАЛОМ е предназначена за Правец-8Д. Написана е на Бейсик с включена машинна подпрограма, извършваща скрол на екрана. Целта на играта е да се преодоляват препятствията по пътя на скиорчето — вратите и борчетата. Движението на слаломиста се осъществява чрез стрелките. Спускането се извършва в три манша — според възможностите на играча.

При оформлението на играта са използвани възможностите на компютъра за псевдографика. Прекодирани са знаковете от латинската азбука. Използвани са и звукови ефекти, но без да се прекалява с тях. Движенията на фигурките са сравнително плавни и не дразнят окото на играещия. Предвидено е класиране на най-добрите участници в играта.

СТЕФАН МИНЕВ

```
10 CLS:PAPER4:INK3:DIM DD$(12),MM(12)
20 PRINT@13,5;CHR$(20);CHR$(4);CHR$(27);"JC A A A C M";CHR$(27);"H"
30 PRINT@3,16;CHR$(4);"ИГРАТА СЕ ПРОВЕЖДА В ТРИ МАНША"
35 PRINT@3,17;CHR$(17);CHR$(6);"ПРИЯТНИ ЗАНИМАНИЯ !"
40 DATA1,3,7,15,31,63,63,49,1,1,1,1,1,1,1,1,32,48,56,60,62,63,63,35,32
50 DATA32,32,32,32,32,32,32,0,0,0,0,1,3,3,1,1,3,7,15,31,63,63,63,32,48
55 DATA56,60,62,63,63,63,0,0,0,32,48,56,56,48,0,0,0,0,1,3,7,3,7,15,31,63
60 DATA63,63,63,63,60,62,63,63,63,63,63,63,0,0,0,32,48,56,60,56,12,30,30
65 DATA12,30,63,45,45,45,45,12,12,12,12,12,12,0,0,0,0,0,0,0,0,63,63,63
70 DATA63,63,63,63,63,6,9,16,33,33,16,8,4,14,17,32,1,2,0,0,12,0,24,36,2
75 DATA1,1,1,1,18,32,33,34,16,8,4,3,18,33,0,32,16,24,36,3,2,8,4,2,2,4,8
80 DATA48,63,63,63,63,63,63,0,0,63,63,1,1,63,63,0,0
90 FOR K=#B60B TO#B6C7:READB:POKE K,B:NEXTK
100 FOR K=#98FF TO#9971:READB:POKE K,B:NEXTK
110 A1$="A"+CHR$(10)+CHR$(8)+"BOD"+CHR$(11)+CHR$(8)+"C"
120 A2$="A"+CHR$(10)+CHR$(8)+"BOOD"+CHR$(11)+CHR$(8)+"C"
130 A4$=CHR$(10)+CHR$(8)+CHR$(8)+CHR$(8)+CHR$(8)+"IJKL"
135 AA$=CHR$(10)+CHR$(8)+CHR$(8)+CHR$(8)+"EJKH"
140 A3$="FG"+AA$+A4$+CHR$(10)+CHR$(8)+CHR$(8)+CHR$(8)+"BD"
145 AA$="QRS"+CHR$(10)+CHR$(8)+CHR$(8)+CHR$(8)+"TUV"
150 A4$="M"+CHR$(10)+CHR$(8)+"N"
155 A9$=" "+CHR$(10)+CHR$(8)+" "
160 A7$=CHR$(10):A8$=CHR$(11):FOR K=1TO9:A7$=A7$+CHR$(8):A8$=A8$+CHR$(8):NEXT
170 A5$="PPPPPPPPP"+A7$+"P ЕТАРТ Р"+A7$+"PPPPPPPPP        PPPPPPPPP"+A8$
180 A5$=A5$+"P ЕТАРТ Р"+A8$+"PPPPPPPPP":A6$="PPPPPPPPP"+A7$+"P ФИНАЛ Р"+A7$
190 A6$=A6$+"PPPPPPPPPOOOOOOOPPPPPPPPP"+A8$+"P ФИНАЛ Р"+A8$+"PPPPPPPPP"
200 DATA8,120,72,138,72,152,72,162,144,173,168,187,201,77,208,2,169,32,201,78
210 DATA240,10,172,128,187,192,78,240,3,141,128,187,24,169,1,109,9,153,141
220 DATA9,153,169,0,109,10,153,141,10,153,24,169,1,109,22,153,141,22,153
230 DATA141,29,153,169,0,109,23,153,141,23,153,141,30,153,236,9,153,208,188
240 DATA162,191,236,10,153,208,179,169,168,141,9,153,169,187,141,10,153,141
250 DATA23,153,141,30,153,169,128,141,22,153,141,29,153,104,168,104,170,104
260 DATA40,96
265 GOSUB1000:EW=0:M1=0
270 PAPER0:INK7:CLS:X=19:X1=19:I=0:F=1:Y=0:M=0:Z=0:POKE#24E,7:POKE#24F,2
271 IF EW=1 THEN PRINT@5,26;"ТОЧКИ: 0                    МАНШ:3";:GOTO275
273 PRINT@5,26;"ТОЧКИ: 0                    МАНШ:1";
275 PRINT@X,5;A4$
277 PRINT@7,10;A5$
280 T$=KEY$:IF T$=CHR$(8) AND X>2 THEN X=X-1
290 IF T$=CHR$(9) AND X<39 THEN X=X+1
295 S=SCRN(X,7):IF X=X1 THEN310
300 PRINT@X,5;A4$:PRINT@X1,5;A9$:X1=X
310 IF S=79 THEN Z=Z+10:F=1:PRINT@12,26;MID$(STR$(Z),2);:GOTO330
320 IF S=32 THEN 330
325 Z=Z-50:PRINT@X-1,5;AA$:IF Z<0 THEN Z=0
327 GOSUB3620:CALL#98FF:WAIT(30):CALL#98FF:WAIT(30):CALL#98FF:WAIT(30)
328 PRINT@X,5;A4$:PRINT@12,26;"    ";:PRINT@12,26;MID$(STR$(Z),2);
329 IF EW=1 THEN M=M+M1:GOTO2000
330 CALL#98FF:I=I+1:IF I>19 THEN I=0
333 IF M<Z THEN M=Z
335 IF Y>99 THEN390
340 IF Z>300 THEN J$=A1$:ELSE J$=A2$
343 IF EW=1 THEN F=0
345 K=INT(20*RND(3)+X-10):IF K>35 OR K<3 THEN345
347 IF I=0 AND F=0 THEN PRINT@2,22;"":PRINT"WXWXWXW
```

Филтър срещу електростатичното поле



Много са изискванията към който и да било съвременен видеомонитор. Едно от тях е екранът да не отразява падащата върху него светлина и да не дава уморяващи зрението отблясъци. За целта върху кинескопа се нанасят специални покрития или се използват отделни филтри, които се поставят пред екрана. Такива филтри отдавна се продават на световния пазар. Новото е, че вече има филтри, които притежават още едно изключително ценно свойство — да екранират създаваното от кинескопа електростатично поле. Неговият потенциал възлиза на около 20 000 V и е доказано, че оказва повече или по-малко, но във всеки случай вредно въздействие върху човешкия организъм. Екраниращото действие се дължи на специален слой, нанесен на задната страна на филтъра.

За широкото разпространение на филтрите говори и фактът, че производителят осигурява 800 различни варианта за съответните типове видеомонитори.

Миниуинчестър

Едва ли има някой, който може да си представи персонален компютър, предназначен за нещо по-различно от игри, който да няма запаметяващо устройство с твърд диск — уинчестър. Колкото компютрите стават по-малки, толкова повече се смаляват и дисковите устройства. Най-строги са изискванията към уинчестърите на преносимите компютри, които освен достатъчно голям капацитет и устойчивост на сътресения трябва да бъдат и колкото е възможно по-малки.

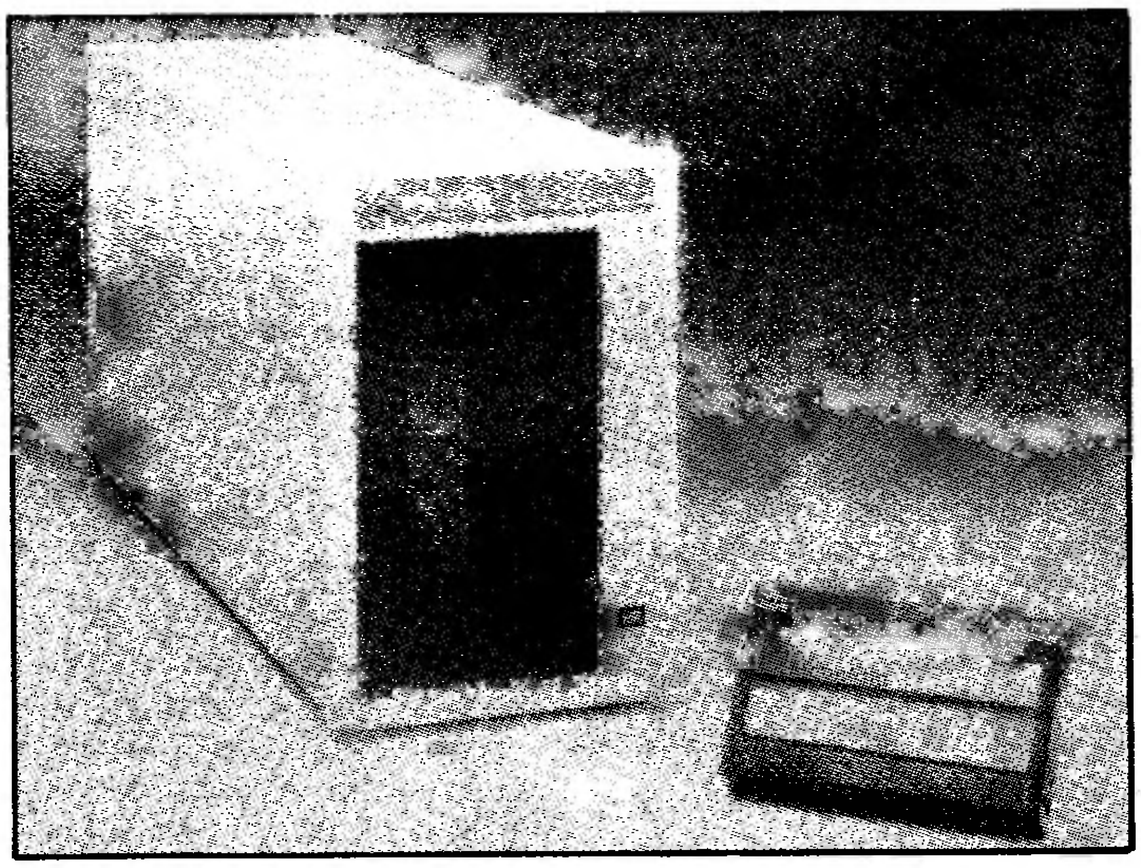
За пръв път границата от един инч (2,54 cm) за височината на дисковото устройство бе достигнатата при преносимия компютър Compaq-Laptops SLT/286. Капацитетът на неговия уинчестър е 20 или 40 Мбайта при средно време на достъп 29 ms.

За големината на миниуинчестърите може да се съди и по снимката, на която друг един модел — LP-HD20 на фирмата ACR е съпоставен с обикновена 5 1/4-инчова дискета.

Колкото по-голям е капацитетът на твърдия диск, толкова повече ценна информация може да се загуби при неговото повреждане или неволно изтриване. Изходът е един — редовно да се прави копие на информацията.

При дисковете с малък капацитет, например 10 Мбайта, (или при липса на друга възможност) може да се използват и дискети. Ако се използва командата BACKUP на DOC за това са необходими около 28—30 дискети и около пововин час време при предварително форматирани дискети. Има и редица специализирани програмни продукти за създаване на резервно копие. В популярния PC TOOLS DELUX е включена програмата PCBACKUP, с която се работи значително по-удобно в сравнение с BACKUP на DOC — менюта за избор на справочниците и съдържащите се в тях файлове, съхранява се списък на записаните файлове, едновременна работа с две флопидискови устройства, възстановяване на отделни файлове и т. н. Още по-съвършен е специализираният продукт FASTBACK PLUS, който е и два пъти по-бърз. В многобрайните му възможности влиза и компресирането на архивираната информация, така че вместо 360 Кбайта на една дискета се

2.2. Гигабайта на касета

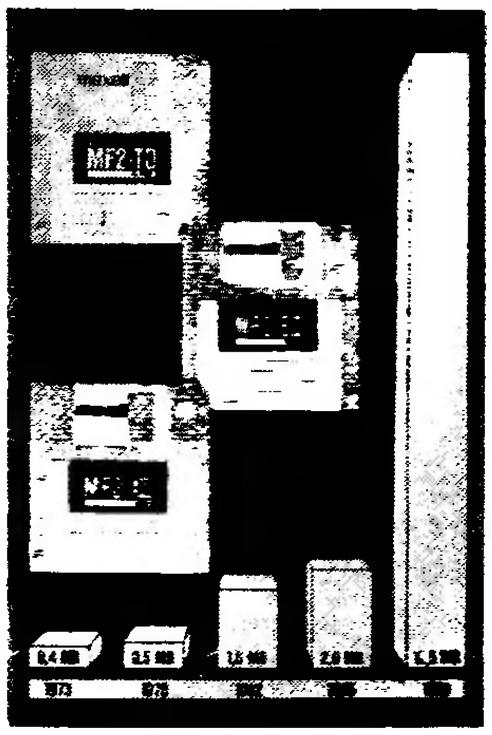


побират около 600 Кбайта (в зависимост от съдържанието на файловете) при два пъти по-ниска скорост на записа — около 500 Кбайта/мин.

За по-големи обеми информация най-доброто средство за създаване на резервно копие са специалните записващи устройства с магнитна лента. На снимката е показан модел на фирмата Mountain Computer, който записва от 40 Мбайта до 2.2 Гбайта на касета с осемилиметрова лента. Скоростта на записа е 13 Мбайта на минута. Програмното осигуряване на записващото устройство гарантира много висока сигурност на записа. Според производителя е възможна една некоригирана грешка на двеста записани касети. Устройството е съвместимо с компютрите IBM AT.

12 Мбайта върху 3 1/2" дискета

В съревнованието за съзнаване на магнитни носители с все по-голям капацитет фирмата MAXELL направи впечатляваща крачка напред, като започна производството на 3 1/2" дискети с капацитет 4 и 12 Мбайта. Ако новината и днес звучи сензационно, само до преди две-три години подобно съобщение би изглеждало немислимо - постигнатият ръст най-добре проличава на диаграмата, която проследява развитието от 1973 година насам.

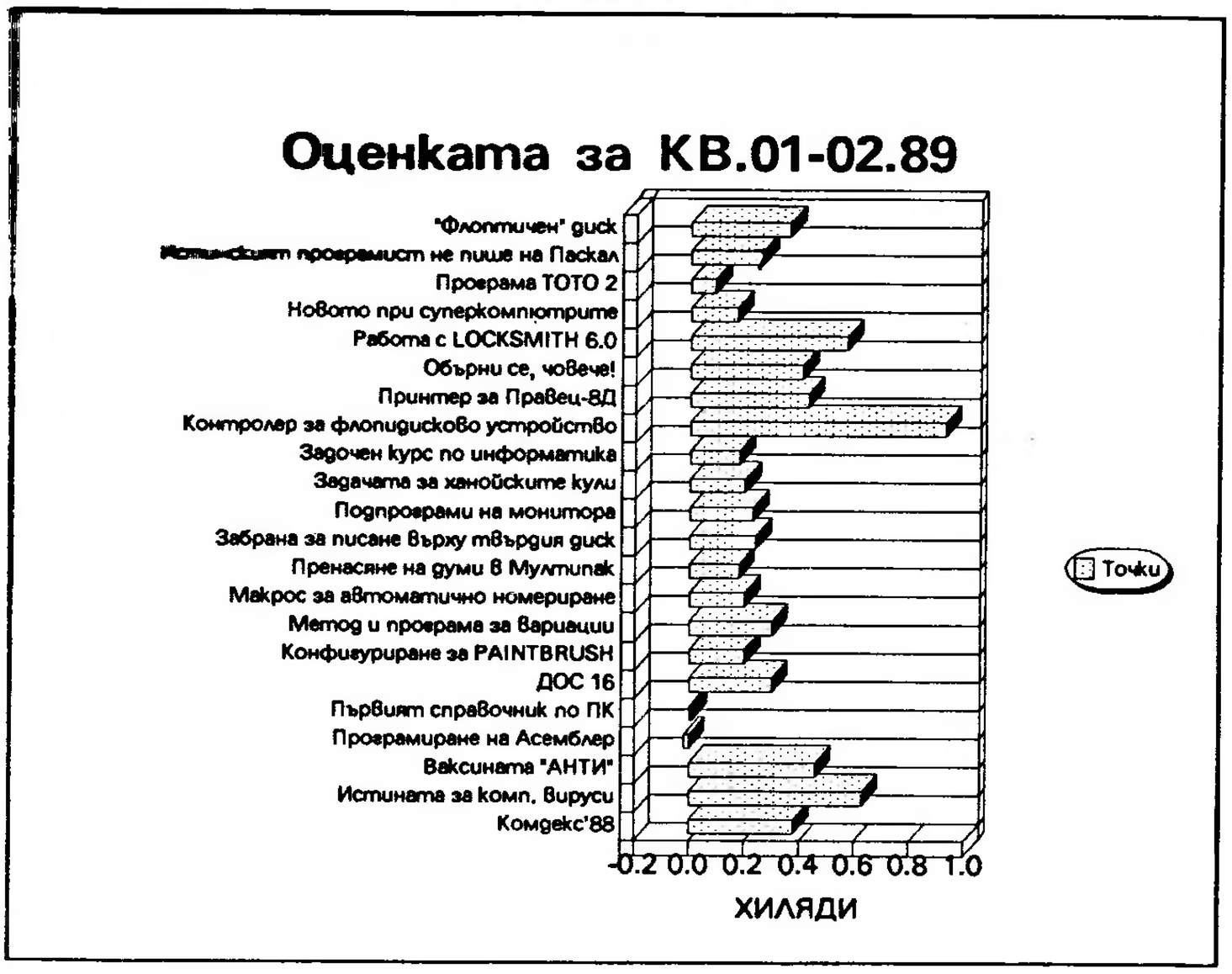


Фирмата NEC вече е създала и съответното 12 Мбайтово дискетно устройство, което засега вгражда в собствена текстообработваща система, а няколко други производители са започнали разработката на 4 Мбайтови дискетни устройства. Новите дискети и дискетни устройства са напълно съвместими с досега произвежданите от същия формат. Това означава, че дискетите със свръхвисока плътност могат да се използват и от досегашните флопидискови устройства, а обикновените дискети могат да бъдат четени от новите дискетни устройства. Разбира се, при пониската плътност на записа.

12,5" Мбайтовите (неформатирани) дискети носят означението MF2-TD и имат по 406 пътечки на инч. MF2-ED е означението на 4 Мбайтовите дискети с по 135 пътечки на инч.

ИЗДАТЕЛ на СО ППС

[illegible] „ОБЕКТИВ" за този брой жребият оп[illegible] „Най-доброто от „Компютър за вас", т.1 да получи [illegible] С. [illegible], Варна, „Македония" 35.



„Най-доброто от „Компютър за вас", т.1

Статистическа обработка на данни с персонален компютър Правец-82

ПРЕМИЯ НА ЧИТАТЕЛЯ

1. Кога медианата и средната стойност имат еднакъв знак?

☐ винаги ■ понякога ☐ никога

2. Ще отхвърлим ли хипотезата, ако статистиката на Стюдънт е равна на 3 при ниво на доверие 95% и четири наблюдения?

☐ винаги ■ понякога ☐ никога

3. Ще се промени ли стойността на хи-квадрат критерия, ако разменим емпиричната и теоретичната функция на разпределение?

■ да ☐ не

4. Когат коефициентът на Пирсън при повече от сто наблюдения е равен на единица?

☐ понякога ■ никога ☐ винаги

5. Разполагате с данните за печалбата на три предприятия за период от пет години. Каква е матрицата от наблюдения?

■ 5 реда, 3 стълба ☐ 15 реда, 1 стълб

■ 3 реда, 5 стълба ■ 15 реда, 3 стълба

Верните отговори са отбелязани с тъмни квадратчета.



Уважаеми читатели,
Отбележете в квадратчетата с 1, 2 или 3 материалите от броя, които класирате съответно на първо, второ и трето място. Ако по ваша преценка някой материал е маловажен или слабо разработен, можете да го отбележите с „Х".
В долните редове напишете какви теми, материали или въпроси бихте желали прочетете в списанието.

ЕГН

Професия

Съдържание

ДРАГИ ЧИТАТЕЛИ 1
ПРЕД ДИСПЛЕИТЕ НА БЪДЕЩЕТО 2
КАКВО ПРАВЯТ КОМПЮТРИТЕ С ДЕЦАТА? 4
ПРАДЯДОТО НА КОМПЮТЪРА 6
„ГОЛЯМАТА КОШНИЦА" Е ПЪЛНА! 7
ОЩЕ ЗА КОМПЮТЪРНИТЕ ВИРУСИ 8
ИЗЧИСЛИТЕЛНИЯТ ПОДХОД В НАУКАТА И ПАРАЛЕЛНИТЕ КОМПЮТРИ 16
IBM PC DOS 4.0 21
„ОБРАЗЕЦ 1" ЗА УЧИЛИЩЕТО 26
КНИГОПИС 27
ЕЛЕКТРОННА ПОЩА 28
ПРОЗОРЦИ В МИКРОФАЙЛ/16 31
ТОПЪЛ И СТУДЕН РЕСТАРТ 33
РАЗПОЗНАВАНЕ НА МОДЕЛА 34
МОЖЕ ЛИ КОМПЮТЪРЪТ ДА СМЯТА 35
ПОЩА 37
ОПИСАНИЕ НА КОМАНДИТЕ НА CP/M 38
ПОДПРОГРАМИ НА МОНИТОРА 40
COMPILER PLUS 44
РАБОТА С LOCKSMITH 6.0 47
ЗАДОЧЕН КОНКУРС ПО ИНФОРМАТИКА 49
ЧИСЛЕНО РЕШАВАНЕ НА КУБИЧНО УРАВНЕНИЕ 52
ИЗЧИСЛЯВАНЕ НА ФИЛТРИ НА ЧЕБИШЕВ 56
ЕЛЕКТРОНЕН КАЛЕНДАР — БЕЛЕЖНИК 57
СЛАЛОМ 60
ПАНОРАМА 62

ДОПЪЛНЕНИЕ

Преди стартиране на програмата „МОНИТОР ЗА ПРАВЕЦ–8Д" на Борислав Захариев (КВ. 09–10.88) трябва да се въведе в директен режим НІМЕМ # 9000.

КОМПЮТЪР ЗА ВАС

Издание на ЦК на ДКМС

Главен редактор:
Георги Балански
87-09-14

Дежурен редактор:
Анелия Ерменкова

Дизайнер:
Васил Пендев

Технически редактор:
Люба Калпакчиева

Коректор:
Румяна Попова

1000 София
бул. „Толбухин" № 51 А
87-50-45

За публикуване се приемат програми, записани на магнитен носител — дискета или касета, които редакцията връща обратно. Ръкописи не се връщат.

Предадено за печат

Подписано за печат

ПЕЧАТНИ КОЛИ 8

Формат 60/90/8

Тираж 30 000

Цена 1,60 лв.

ДП „Д. Благоев" 2
София, ул. „Ренетин" 2
Телефон 46–31

РЕДАКЦИОНЕН СЪВЕТ: ст.н.с. инж. Александър П. Александров, чл.–кор. Ангел Ангелов, проф. Ангел Писарев, акад. Благовест Сендов, Веселин Спиридонов, доц. Димитър Шишков, инж. Иван Марангозов, инж. Иван Михайлов, доц. к.т.н. Лазар Лазаров, Николай Боев, инж. Пенчо Сираков, чл.–кор. Петър Кендеров, инж. Петър Петров, ст.н.с. к.т.н. инж. Пламен Вачков, Рашко Ангелинов, инж. Стефан Чернев

Индекс 20 593

Текстообработваща програма

ПРОТЕКСТ

Справочник на командите

Основни команди

?	Извиква справочник за командите (от помощното меню)
RETURN	Преминаване в режим на въвеждане на текст
RETURN	Завършване на абзац
Осв	Първата написана буква е главна
Осв Осв ^	Връщане в помощното меню
МК-Ь	Превръщане на редовните в главни букви и обратно
МК-В	Разделяне на екрана на две части
МК-Г	Писане само с главни букви
МК-А	Превключва кирилица/латиница
МК-Ж	Превключва режим на вмъкване/заместване на текст
МК-Э	Въвеждане на управляващи знакове

Зареждане на файлове

МК-Ч:име файл [,Dn]	Зареждане на файл (текст) от дискетата
МК-Ч:? [,Dn]	Показва каталога на дискетата за избор на файл
МК-Ч:=	Зарежда файл с името показано в информационния ред
МК-Ч:име файл,/маркер1/маркер2/	Зарежда текста между маркерите включително
МК-Ч:име файл,/маркер/	Зарежда текста от маркера до края на текста
МК-Ч:име файл,//маркер/	Зарежда текста от началото на текста до маркера
МК-Ч:име файл,/маркер1/маркер2/А	Зарежда всички сегменти, намиращи се между маркерите
МК-Ч:име файл,/маркер1/маркер2/Н	Зарежда текста между маркерите (без тях)
МК-Ч:%/маркер1/маркер2/	Копира текста (от паметта) между маркерите на позицията на курсора
МК-^:име файл,Э	Показва съдържанието на файла на екрана без да го зарежда в паметта (МК-Я спира преместването на текста)

Записване на текстове

МК-З:име[,Dn]	Записва текста във файл на дискета
МК-З:=	Записва текста с името записано в командния ред
МК-З:име+	Добавя текста от паметта към вече съществуващ файл
МК-З:име/маркер/	Записва частта от позицията на курсора до маркер
МК-З:име/маркер/+	Добавя маркирана част от текста към вече съществуващ файл
МК-З:?	Показва каталога на дискетата

Придвижване на курсора

Осв Осв	Преминаване в режим за придвижване на курсора

Клавиши

Т,Н	Придвижват през един знак наляво/надясно
С,П	Придвижват през един ред нагоре/надолу
Е,Й	Придвижват през 12 реда нагоре/надолу
Я,А	През 24 знака или до интервал наляво/надясно
Друг клавиш	Връщане в режим на въвеждане на текст
МК-Н	Придвижване в началото на текста (поставя показалеца за посока)
МК-К	Придвижване в края на текста (поставя показалеца за посока)

Следва


PRAVEC