Тажимамат уулу Кубанычбек

Компьютер менен ырыскы табуу жолдору жана программирование боюнча устачылык

Ош шаары

26 раби уль аваль, 1428 ж

БисмиЛляхи Рахмани Рахим.

Бул дүнүёдө баарына, ишенгендерге дагы, ишенбегендерге дары ырыскыны
бергенге кудрети жетүүчү
жана кыямат куну мусулмандарга кечиримдүү
Алланын аты менен баштайм.

Ассаламу алейкум, урматту окурман.

Элибиздин абалы оорлошуп кетти. Коп жыл мурун биз чоң мамлекеттин атуулдары болчу элек. Эми тырмактай мамлекет болуп калгандан кийин чоң мамлекеттер менен тышкы соода жана саясат иштерин жүргүзүш таптакыр татаал болуп калды. Чоң мамлекеттен калган өндүрүш талкаланды, айыл чарбанын түшүмүнүн сыртка сатыла турган баалары биз үчүн дагы өтө төмөн болуп калды. Ошол эле убакытта биз сырттан түшүмү жогору чарбалардан чыгарылган нерселерди сатып алабыз. Бул абалды кандай оңдосок болот? Туура, эл даражасында бул баарыбыздын ишибиз, бирок азыр күнүмдүк каражатты кандай жол менен тапканыбыз оң? Муруңку марксизм түшүнүктөрү боюнча ыктысад (экономика) өнүгүшү үчүн акча-каражат керек дейт. Бирок өзүбүздүн Ислам боюнча ыктысад элдин пикир денгээлине карай өсөт. Эгер биз терең пикирлесек, кандай чарба биз үчүн түшүмдүү болот?

Биз колдон келишинче чыгашасы аз бирок кирешеси көп чарбаны негизги улуттук чарба катары тандаш керекбиз. Мисалы Суоми (Финляндия) үчүн бул телефондорду жасап чыгаруу болду. Айыл чарба менен тоо кен чарбалар сырттагы бааларга көз карандуу. Тоо кен, жеңил өнөр-жай жана оор өнөр-жай үчүн мамлекеттен уруксат алыш жана салык төлөш өтө татаал.

Эгер биз дүнүёдөгу абалды карасак акыркы убакытта компьютерлер менен байланышкан өндүрүштөр өтө тез өнүгүп жатат. Компьютер жеке өзу анча көп пайда алып келе албайт, бирок анын күчү менен башка өндуруштөр үчүн көп мурунку кыйынчылыктар чечилип жатат. Ал жумушту жакшы билген адамдар чоң мамлекеттерде өздөрүндө жетишпей, бизге окшогон мамлекеттерден ишке көзү жеткендерди терип алды. Бирок Индия жана Россияга окшоп биз өзүбүздүн ишке жараган адамдарыбызды сыртка чыгарып койсок алар жумушу менен өз элине эмес, башка элге көбүраак пайда алып келет.

Демек, биз башка элде ушундай иштерге мүмкүнчүлүктөрдү издеп, өзүбүздүн элибизди өзүбүздүн жерибизде иштетиш керекбиз. Муну кандай жол менен кылса болот? Азыр интернет деген тармак бар, бизде дагы жакшы өнүктү. Эл үйүнөн алыс эмес жерде компьютер менен (кыйналбай) иш кылып иштин жыйынтыгын интернет аркылуу жөнөтүп жиберсе болот. Бажыканадан өткөзөм, пара берем, каракчыларды багам, жол кире табам деген маселелер жок.

Бул нерсени ишке ашырыщ үчүн

биринчиден сыртта ишкана болуш керек кардарларды жана мүмкүнулүктөрдү издеген,

экинчиден эл керектүү жумушту компьютер менен аткарганды билиш керек.

Биринчисин бизге бериңиздер, экинчисин бирге кылалы. Биздин элде башка элдерге караганда бир чоң артыкчылык бар. Мисалы үчүн Индия элинин көбү окуганды жана жазганды билбейт. Ошону карабастан алар компьютер менен сырттагы кардарлар

үчүн өтө көп жумуш жасайт. Түшкөн пайдасы Россиянын сыртка курал саткан пайдасынан көбүраак. Биз эмне үчүн элибиз окуганды – жазганды билгенин пайдаланбайбыз? Демек, кандай жол менен элди жумушка даярдасак болот?

Биринчиден, бул китепте мен билген нерсени жазганы аракет кылдым. Бул маалыматты мен көп жылдан бери топтодум. Балким бул китепти окуу жайларда дагы окуу курал үчүн падаланса болот аты туура келген сабактарда.

Экинчиден, “сетевой маркетинг” деген балекет чыкты. Анын өзүн мен көп жактырбаган менен бир мүмкүнчүлүгүн көрдүм. Аз убакытта көп адамды жумушка киргизсе болот экен. Алардын кээбир усулдарын колдонуп, бирок өтпөгөн товарын элге саттырбай, компьютерде натыйжалуу иштегенди ошол жол менен үйрөтсөк болот. Бирөө 2 адамды үйрөтсө, алар 4 адамды үйрөтсө, алар 8 адамды үйрөтсө, алар 16 адамды үйрөтсө, алар 32 адамды үйрөтсө, алар 64 адамды үйрөтсө, алар 128 адамды үйрөтсө, алар 256 адамды үйрөтсө, алар 1024 адамды үйрөтсө, алар 2048 адамды үйрөтсө, алар 4096 адамды үйрөтсө, алар 8192 адамды үйрөтсө, алар 16384 адамды үйрөтсө, алар 32768 адамды үйрөтсө, алар 65536 адамды үйрөтсө, алар 131072 адамды үйрөтсө, алар 262144 адамды үйрөтсө, алар 524288 адамды үйрөтсө, алар 1048576 адамды үйрөтсө. Жыйырма кадамдын ичинде миллион деген санга жетсе болот экен.

Сырттагы ишкана баарына жетишкидей жумуштун көлөмүн даярдап туруш керек. Компьютер менен жумушту чоң кишилер гана эмес, балдар дагы аткарса болот. Кыйналып араба түртүп бөөрөктөрүнө суук тийгизгендин оордуна отуруп, ойлонуп, ойлорун компьютер менен ишке ашырып ошол жол менен ырыскы тапкан жакшы эмеспи? Анда эмесе урматту окурман, сабакты баштайлы. БисмиЛляхи Рахмани Рахим.

Компьютер кандай иштейт?.

Көпчүлүгүбүз күйүүчү май менен жүргөн машинелер кандай иштейт экенин жакшы түшүнөбүз. Компьютер менен биз көргөнбүз, ар нерсе кылынат: сүрөттү көрсө, обондорду укса, кинону көрсө, каттарды жана китептерди жазса, чийме чийсе, оюн ойносо, эсеп чыгарса болот. Бул нерселер компьютердин ичинде кандай жол менен кылынат? Эгер аны биз туура түшүнсөк компьютерди натыйжалуу пайдалансак болот.

Биричиден биз эске салыш керекбиз, компьютер маалымат менен иштейт: аны сактайт жана берилген эрежелер менен өзгөртөт.

Экинчиден компьютердин акылы жок, акылды Кудая Таала адамзатка берген, адамзат өзу башка акылды жарата албайт канча аракет кылса дагы. Аны карыдин адамдар түшүнбөй “жасалма акылды” түзгөнгө (искусственный интеллект, artificial intellect) жарым кылымдан бери натыйжасыз бекер аракет кылып жатат. Демек, эгер маселени чечкенге биз анык эрежелер боюнча аткарала турган буйруктарды бере алсак гана биз ошол маселени компьютерге чечтире алабыз. Ал буйруктарды Аль Хорезми аалымдын атынан латын тилинде окулушу боюнча АЛГОРИТМ деп атады.

Үчүнчүдөн, компьютер жиндердин же сыйкырдын күчү менен иштебейт, аны бизге окшогон адамдар акыл жолу менен жасап чыккан заттардын табийгый сапаттарын пайдаланып.

Демек, кандай заттардын сапаттарын иштетет экен?

Биринчиден, ар бир маалыматты биз белги менен белгилесек болот. Ошондо эң аз пайдаланган белгилердин саны эки болот экен: же БАР, же ЖОК. Ага башка маанилерди берсек болот: ӨЧТҮ-ЖАНДЫ, КЕЛДИ-КЕТТИ, ЧЫН-КАЛП. Эсиңиздеби, телефон менен башка адамдардын көзүнчө сүйлөшкөндө сырды билгизбеш үчүн нары жактан сизге толук сөздөрдү айтат, сиз болсо айтылган жөнөкөй жоопторду берсеңиз болот. Эгер көбүраак

маалыматты берем десеңиз аны көбүраак ошондой жөнөкөй белгилер менен белгилесеңиз болот, тилин түшүнсөңүз эле болду.

Экинчиден, ошол белгилерди сактай турган нерсе керек экен. Дубалды карап көрсөңүз, өчүргүч турат, ал лампочкада нурду жандырат же өчүрөт бир басым менен. Мына, ошол нерсеге маалымат сактаса болот. Мисалы үчүн, эгер үйдүн ээлери үйдө жок болуп түрса нурду жандырып коёт: бул ууруларга белги, үйдө ээси бар болуш мүмкүн деген, аны көрүп уурулардын көбү коркуп үйгө кирбей турат. Демек, уурулар анын тилин түшүнөт экен.

Комптьютердин бөлчөктөрүндө бул абалдар ар кандай жолдор менен сакталат. Тарыхта эң биринчи перфокарталар колдонулган. Перфо деген тешик. Катуу кагазда кээбир жерлери бүтүн, кээбир жерлери тешик болуп жасалган. Ал катуу кагаз эки электродтун ортосунда турат. Тешик бар болсо электродтор биригип токту өткөрөт. Болбосо жок.

Башка мисал – кээбир магниттер электр талаанын тасиринен магниттик багытын өзгөртө алат. Ал багытка ошол эле эки абалды сактаса болот.

Дагы бир мисал – компакт дисктерде күзгү жана күзгү эмес жерлери бар. Күзгүдөн лазер нуру чагылат, күзгү эмес жерлерге сиңип кетет. Ошентип эки түрлүү абалдарды сактаса болот.

Эң маанилүү мисал физиканы билгендер үчүн – бул конденсатор. Ичинде чыналуу же бар же жок. Бул жол менен эң тез түрүндө эки абалды сактап жана өзгөртсө болот.

Ал эми эки абалды сактай алсак андан көп абалдарды дагы сактасак болот. Кантип? Сактагычтардын санын көбөйтүп. Бир абалга «0» деген атты берели, башкасына «1» деген атты берели. Бир сактагычка «бит» денег атты берели. Англис тилинен бул сөз «бир кичинекей үзүм» деп которулат. Сактагыч бирөө болсо, ага эки айткандай абалды сактасак болот:

0, 1.

Сактагычтан экөө болсо төрт абалды сактасак болот:

00, 01, 10, 11.

Катарды эстеп калыңыз. Жанаша турган эки сактагычтын башындагы битке «улуу бит» наамды берели, аягында турган битке «кенже бит» деген наамды берели (Сактагычтардын саны көбөйсө «улуу биттер» дана «кенже биттер» денег сөздөргө таңгалбагыла). Өзгөрүү кенже биттен башталып улуу битменен аяктайт. Эгер айтылган эреже менен үч сактагычты көрсөк, аларда сегиз абалды сактаса болот:

000, 001, 010, 011, 100, 101, 110, 111.

Көрдүңөрбү. Сактагычтардын саны бир аз көбөйсө алар сактай турган абалдардын саны кыйла эле көбөйөт. Илгери бир жомок бар болчу эле. Шахматты ойлоп тапкан адам акысына ошого окшош жол менен шахмат тактасынын 64 жайына дан толтуруп бергиле деген. Биринчи жайга 1 данды, экинчи жайга 2 данды, үчүнчү жайга 4 данды, ошентип ар бир кийинки жайга эки эсе көбүраактан сураган. Көрсө тактанын жайлары түгөгүчө жер жүзүндөгү бүткүл дан азык калбай калат экен. Компьютердин күчү мына ушунда. Сактагычтардын саны анча көп эмес көбөйсө, алар сактай турган маалымат өтө көп көбөйөт. Өзүңөрдү сынаш үчүн абалдарды жазып көргүлө. Мен натыйжасын чыгарып берейин.

Сактагычтардын саны	Абалдардын саны
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768
16	65536
17	131072
18	262144
19	524288
20	1048576
21	2097152
22	4194304
23	8388608
24	16777216
25	33554432
26	67108864
27	134217728
28	268435456
29	536870912
30	1073741824
31	2147483648
32	4294967296

Азыркы компьютерлер бир иштегенде 32 зым аркылуу иштейт (бир «көз ирмесинде» 32 бит аркылу маалыматты берет, бул төрт миллиардтан көп ар кандай абалдар). Кол менен дептерге жазып чыгам десеңер бир абалды 1 мүнөттүн ичинде жазсаңар күнүнө сегиз сааттан иштеп, бул ишти саатты 60 мүнөткө бөлсөк, күнүдү 8 саатка бөлсөк, жылды 365 күнгө бөлсөк 1024 жылдын ичинде жазып чыкса болот экен. Баракелде. Ушундай пайдасыз жумушту компьютер адамдан канча эсе тезираак аткарат экен. Эми сиздин максатыңыз, урматту окурман, компьютердин күчүн билип ага пайдалуу жумушту кылдыруу.

Көп абалдар туурасында маалыматты сактай алат экенбиз. Ал эми ошол абалдарга маани беришти үйрөнөлү. Ойго эң биринчи келе жаткан маалымат – басма сөз. Ал тамгалардан жана үтүр-чекиттерге окшогон белгилерден түзүлөт. Эгер биз сегиз бит менен чектелсек ал сегиз бит 256 абалды түзө алат. Бул абалдар ар бир чоң жана кичинекей тамгалар менен кошумча белгилерди белгилегенге жетишкидей. Мисалы үчүн биричи абалга «А» тамгасын белгилесек, экинчиге «Б» деп. Сегиз битти «байт» дейли. Ошентип байттар менен биз каалаган басма сөздү жазып чыксак болот экен. Аны биз компьютердин ичине сактай алабыз, башка бирөөгө оңой жол менен көчүрүп бере алабыз жана алыс жерге Интернет байланыш тармагы аркылуу тез арада жөнөтө алабыз. Бул нерсе соодагерлер үчүн, саясатчылар үчүн дана куралдуу күчтөр үчүн өтө пайдалуу жана

маанилүү. Бул нерсени жакшы түшүнүп пайдаланса башка болуп жаткан иштерди тездететурган жолдорун тапса болот.

Кодирование алфавитов разных языков.

15 разных кодировок русского языка

«Умом Россию не понять,
Аршином общим не измерить,
У ней особенная стать,
В Россию можно только верить» (но современная деидеологизированная Россия не верит уже в саму себя и не знает, по какому светлому пути идти, погрязнув в потребительстве, ей иногда не нравится западная идея секуляризма, но от нее она уже не может отказаться, а ее лозунг «Россия для русских» не сможет ее укрепить и сохранить, так как украинец и татарин не захотят стать русскими)

Кодирование чисел целых (беззнаковых, знаковых) и вещественных
Кодирование графических объектов (чертежей, фотографий)
Как Рене Декарт избавил при помощи своей системы координат театры от кровавых дуэлей (Европейские театры – это продолжение языческих мистерий и оргий древнего Рима).
Кодирование звука

Простейшие битовые операции
Машинные коды
Язык высокого уровня и функции компилятора

4 фактора, необходимых для мышления:
здоровый человеческий разум,
объективная реальность,
органы чувств, передающих информацию от бытия к мозгу ,
предварительная информация.

Коммунисты правильно сказали, что бытие определяет сознание, но они неправильно понимали третий и четвертый пункты, думали, что мышление это отражение бытия в мозге и поэтому поверили в недоказуемое и неосязаемое – в вечность мира.

Компьютерге буйрук берүнүн негиздери.

Компьютерге буйрук берүү деген бул бир бүтүн нерсени бөлүкчөлөрдөн топтоо, имаратты курганга окшош.

Минимально необходимый синтаксис Компонентного Паскаля

Маалыматты тартипке салуу

В книгу добавить главу о генерации исходного кода модуля,
компиляции и запуске процедур из него на лету,

использование в работе компилятора
(конфигурационные настройки,
анализ символьных файлов)

Компилятордун ичи

жана метапрограммированиени колдонуу мисалдары.
Метапрограммирование на уровне исходных текстов, сравнение с Си и Си++
Метапрограммирование на уровне скомпилированных модулей

----------Пример модуля, генерирующего код на основе данных компилятора------

```
MODULE Robot;
(**
	для того, чтобы принять новый модуль данных, надо выгрузить этот модуль

	для запуска
	скомпилировать этот модуль,

	открыть модуль BB\System\Mod\Config.odc

	добавить строку
	Converters.Register("Robot.ImportSymFile", "", "TextViews.View", "osf", {});

	рядом с похожими, скомпилировать и перезапустить

	Скомпилировать модуль Totalfem/Mod/Xl.odc

	и открыть через "Файл-открыть" с типом Robot.ImportSymFile *.osf
	файл Totalfem/Sym/Xl.osf
**)

	IMPORT
		Kernel, Files, Fonts, Dates, Ports, Stores, Views, Properties, Dialog, Documents,
		TextModels, TextMappers, TextRulers, TextViews, TextControllers, StdDialog, StdFolds,
		DevCPM, DevCPT, HostRegistry,
		con:=StdLog, Strings
		;

	CONST
		Undefined=-1;
		Write=1;
		Read=2;

		width = 170 * Ports.mm; (* view width *)
		height = 300 * Ports.mm;

		(* visibility *)
		internal = 0; external = 1; externalR = 2; inPar = 3; outPar = 4;

		(* object modes *)
		var = 1; varPar = 2; const = 3; field = 4; type = 5; lProc = 6; xProc = 7; cProc = 9; iProc = 10;
		mod = 11; head = 12; tProc = 13; ptrType = 31;

		(* structure forms *)
		undef = 0; bool = 2; sChar = 3; byte = 4; sInt = 5; int = 6; sReal = 7; real = 8;
		set = 9; sString = 10; nilTyp = 11; noTyp = 12; pointer = 13; procTyp = 14; comp = 15;
		char = 16; string = 17; lInt = 18;

		(* composite structure forms *)
		basic = 1; array = 2; dynArray = 3; record = 4;

		(* attribute flags (attr.adr, struct.attribute, proc.conval.setval) *)
		newAttr = 16; absAttr = 17; limAttr = 18; empAttr = 19; extAttr = 20;

		(* additional scanner types *)
		import = 100; module = 101; semicolon = 102; becomes = 103; comEnd = 104;

		modNotFoundKey = "#Dev:ModuleNotFound";
		noSuchItemExportedKey = "#Dev:NoSuchItemExported";
		noSuchExtItemExportedKey = "#Dev:NoSuchExtItemExported";
		noModuleNameSelectedKey = "#Dev:NoModuleNameSelected";
		noItemNameSelectedKey = "#Dev:NoItemNameSelected";

		regKey = "Dev\Browser\";
```

```
	TYPE
		TProcList = POINTER TO RECORD
			fld: DevCPT.Object;
			attr: SET;		(* newAttr *)
			next: TProcList
		END;


		tObjectList*=POINTER TO RECORD
				val*:DevCPT.Object;
				next*:tObjectList;
		END;

	VAR
		global: RECORD
			hints: BOOLEAN; (* display low-level hints such as field offsets *)
			flatten: BOOLEAN;		(* include fields/bound procs of base types as well *)
			hideHooks: BOOLEAN;	(* hide hook procedures and interfaces *)
			clientinterface, extensioninterface: BOOLEAN;
			sectAttr, keyAttr: TextModels.Attributes;	(* formatting of section and other keywords
*)
			level: SHORTINT;		(* current indentation level *)
			gap: BOOLEAN;	(* request for lazy Ln, issued with next Indent *)
			singleton: BOOLEAN;	(* selectively display one object's definition *)
			thisObj: DevCPT.Name;	(* singleton => selected object *)
			out: TextMappers.Formatter;	(* formatter used to produce definition text *)
			systemUsed: BOOLEAN;
			comUsed: BOOLEAN;
			modUsed: ARRAY 127 OF BOOLEAN;
			pos: INTEGER
		END;
		inp: Files.Reader;
		imp: ARRAY 256 OF Kernel.Name;
		num, lev, max: INTEGER;

		options*: RECORD
			hints*: BOOLEAN;		(* display low-level hints such as field offsets *)
			flatten*: BOOLEAN;	(* include fields/bound procs of base types as well *)
			formatted*: BOOLEAN;
			shints, sflatten, sformatted: BOOLEAN	(* saved values *)
		END;

		OriginalModuleName:ARRAY 256 OF SHORTCHAR;
		rootCount:INTEGER;
		root:tObjectList;

		mode:INTEGER;
		stackCount,stackMax:INTEGER;
		stack:ARRAY 256 OF DevCPT.Object;
		stackArrayFlag:ARRAY 256 OF BOOLEAN;

	PROCEDURE AppendUniqueObject* (VAR	first:tObjectList; val:DevCPT.Object;		VAR
count:INTEGER);
	VAR
			chain,last:tObjectList;
			found:BOOLEAN;
	BEGIN
				IF	first=NIL	THEN
					NEW(chain);
					chain.val:=val;
					first:=chain;
					count:=1;
				ELSE
					chain:=first;
					found:=FALSE;

					WHILE chain#NIL DO
						IF ~found THEN
							found:=(chain.val.name$=val.name$);
						END;

						IF chain.next=NIL THEN
							last:=chain;
```

```
                                        END;

                                        chain:=chain.next;
                                END;


                                IF ~found  THEN
                                        NEW(chain);
                                        chain.val:=val;
                                        INC(count);
                                        last.next:=chain;
                                END;

                        END;

        END AppendUniqueObject;


        PROCEDURE ^ Int (i: INTEGER);
        PROCEDURE ^ String (s: ARRAY OF SHORTCHAR);
        PROCEDURE ^ Ln ();
        PROCEDURE ^ Tab ();

        PROCEDURE ^ScanPointer (ptr:DevCPT.Struct);
        PROCEDURE ^ScanDynArray (curr:DevCPT.Object);

        PROCEDURE IntToSString* (x: LONGINT; OUT s: ARRAY OF SHORTCHAR);
        CONST
                minLongIntRev = "8085774586302733229";
        VAR j, k: INTEGER; ch: SHORTCHAR; a: ARRAY 32 OF SHORTCHAR;
        BEGIN
                IF x # MIN(LONGINT) THEN
                        IF x < 0 THEN s[0] := "-"; k := 1; x := -x ELSE k := 0 END;
                        j := 0; REPEAT a[j] :=SHORT( CHR(x MOD 10 + ORD("0"))); x := x DIV 10; INC(j)
UNTIL x = 0
                ELSE
                        a := minLongIntRev; s[0] := "-"; k := 1;
                        j := 0; WHILE a[j] # 0X DO INC(j) END
                END;
                ASSERT(k + j < LEN(s), 23);
                REPEAT DEC(j); ch := a[j]; s[k] := ch; INC(k) UNTIL j = 0;
                s[k] := 0X
        END IntToSString;

        PROCEDURE Familia ();
        VAR  familia:ARRAY 256 OF SHORTCHAR;
                        i:INTEGER;
                        buff:ARRAY 256 OF SHORTCHAR;
        BEGIN
                familia:='m.';
                FOR  i :=1  TO stackCount-1   DO
                        familia:=familia+stack[i].name;
                        IF stackArrayFlag[i]    THEN
                                IntToSString(i-1,buff);
                                familia:=familia+"[i"+buff+"]"
                        END;

                        familia:=familia+"."
                END;

                String(familia)
        END Familia;

        PROCEDURE ScanRecord (parent,rec:DevCPT.Object);
        VAR
                curr,curr2:DevCPT.Object;
                found:BOOLEAN;
        BEGIN
                INC(stackCount);
                stack[stackCount-1]:=parent;


                CASE  mode  OF
                        Write:
```

```
                        con.Ln;con.String("ScanRecordWrite "); con.Int(stackCount); con.Ln;

                        IF rec#NIL THEN
                                curr:=rec;

                                WHILE (curr#NIL) DO
                                        IF (curr.typ.form=pointer) &
(curr.typ.BaseTyp.comp=dynArray) THEN

                                                Tab;Tab;String('(* dyn array '+curr.name+' *)');Ln;

                                                curr2:=rec; found:=FALSE;
                                                WHILE (curr2#curr) DO
                                                        IF (curr.name+'Count'=curr2.name$) THEN
                                                                Tab;Tab;String('(* dyn array
counter '+curr2.name+' *)');Ln;
                                                                found:=TRUE;

                                                                IF curr2.typ.form=int THEN

                                                                        Tab;Tab; String('IF
');Familia;String(curr.name+'#NIL THEN ');Ln;
                                                                        Tab;Tab;Tab;Familia;

        String(curr2.name+':=LEN(');
                                                                        Familia;
                                                                        String(curr.name+')');Ln;

                                                                        Tab;Tab; String('ELSE
');Ln;

        Tab;Tab;Tab;Familia;String(curr2.name+':=0');Ln;
                                                                        Tab;Tab; String('END;
');Ln;
                                                                ELSE

                                                                        String('ERROR: counter is
not INTEGER');Ln;

                                                                END;

                                                        END;
                                                        curr2:=curr2.link
                                                END;
                                                IF ~found THEN
                                                        String('ERROR: counter is not
present before dyn array');Ln;
                                                END;
                                        END;

                                        curr:=curr.link
                                END;


                                curr:=rec;

                                WHILE (curr#NIL) DO
                                        con.Tab;con.String(curr.name$);
                                        CASE curr.typ.form OF
                                                bool:
                                                        Tab;Tab;
                                                        String("wr.WriteBool(");
                                                        Familia;
                                                        String(curr.name+");");Ln;

                                                |char:
                                                        Tab;Tab;
                                                        String("wr.WriteChar(");
                                                        Familia;
                                                        String(curr.name+");");Ln;

                                                |sInt:
                                                        Tab;Tab;
                                                        String("wr.WriteSInt(");
```

```
                Familia;
                String(curr.name+");");Ln;

        |int:
                Tab;Tab;
                String("wr.WriteInt(");
                Familia;
                String(curr.name+");");Ln;

        |sReal:
                Tab;Tab;
                String("wr.WriteSReal(");
                Familia;
                String(curr.name+");");Ln;

        |real:
                Tab;Tab;
                String("wr.WriteReal(");
                Familia;
                String(curr.name+");");Ln;

        |pointer:
                IF  (curr.typ.BaseTyp.comp=dynArray)
THEN
                        Tab;Tab;
                        String('FOR i');
                        Int(stackCount-1);
                        String(':=0 TO ');
                        Familia;
                        String(curr.name+"Count-1
DO");Ln;

                        CASE
curr.typ.BaseTyp.BaseTyp.form  OF

                        bool:
                                Tab;Tab;Tab;
                                String("wr.WriteBool(");
                                Familia;
                                String(curr.name+"[i");
                                Int(stackCount-1);
                                String("]);");Ln

                        |char:
                                Tab;Tab;Tab;
                                String("wr.WriteChar(");
                                Familia;
                                String(curr.name+"[i");
                                Int(stackCount-1);
                                String("]);");Ln

                        |sInt:
                                Tab;Tab;Tab;
                                String("wr.WriteSInt(");
                                Familia;
                                String(curr.name+"[i");
                                Int(stackCount-1);
                                String("]);");Ln

                        |int:
                                Tab;Tab;Tab;
                                String("wr.WriteInt(");
                                Familia;
                                String(curr.name+"[i");
                                Int(stackCount-1);
                                String("]);");Ln

                        |sReal:
                                Tab;Tab;Tab;
                                String("wr.WriteSReal(");
                                Familia;
                                String(curr.name+"[i");
                                Int(stackCount-1);
                                String("]);");Ln
```

```
                                                                |real:
                                                                        Tab;Tab;Tab;
                                                                        String("wr.WriteReal(");
                                                                        Familia;
                                                                        String(curr.name+"[i");
                                                                        Int(stackCount-1);
                                                                        String("]);");Ln

                                                                |comp:
                                                                        con.Tab;con.String(" comp
");

                                                                        CASE
curr.typ.BaseTyp.BaseTyp.comp OF
                                                                        record:
                                                                                Tab;Tab;String("(*
RECORD "+curr.typ.BaseTyp.BaseTyp.strobj.name$+" *)");
                                                                                Ln;


        stackArrayFlag[stackCount]:=TRUE;
                                                                                ScanRecord(curr,
curr.typ.BaseTyp.BaseTyp.link);

                                                                        ELSE
                                                                        END

                                                                ELSE

                                                                END;
                                                                Tab;Tab;
                                                                String('END;');Ln
                                                        END
                                                |comp:
                                                        con.Tab;con.String(" comp ");

                                                        CASE curr.typ.comp OF
                                                        record:
                                                                Tab;Tab;String("(* RECORD
"+curr.typ.strobj.name+" *)");
                                                                Ln;


        stackArrayFlag[stackCount]:=FALSE;
                                                                ScanRecord(curr,curr.typ.link);

                                                        ELSE
                                                        END


                                                ELSE
                                                        con.Tab;con.String(" do define ");
                                                END;
                                                con.Ln;
                                                curr:=curr.link
                                        END;

                                END

                        |Read:

                                con.Ln;con.String("ScanRecordRead "); con.Int(stackCount); con.Ln;

                                IF  rec#NIL  THEN

                                        curr:=rec;

                                        WHILE  (curr#NIL)  DO
                                                con.Tab;con.String(curr.name$);
                                                CASE  curr.typ.form  OF
                                                        bool:
                                                                Tab;Tab;
                                                                String("rd.ReadBool(");
```

```
                Familia;
                String(curr.name+");");Ln;

        |char:
                Tab;Tab;
                String("rd.ReadChar(");
                Familia;
                String(curr.name+");");Ln;

        |sInt:
                Tab;Tab;
                String("rd.ReadSInt(");
                Familia;
                String(curr.name+");");Ln;

        |int:
                Tab;Tab;
                String("rd.ReadInt(");
                Familia;
                String(curr.name+");");Ln;

        |sReal:
                Tab;Tab;
                String("rd.ReadSReal(");
                Familia;
                String(curr.name+");");Ln;

        |real:
                Tab;Tab;
                String("rd.ReadReal(");
                Familia;
                String(curr.name+");");Ln;

        |pointer:
                IF  (curr.typ.BaseTyp.comp=dynArray)
THEN
                        Tab;Tab; String('IF
');Familia;String(curr.name+'Count>0 THEN ');Ln;

                        Tab;Tab;Tab;String('NEW(');
                        Familia;
                        String(curr.name+",");
                        Familia;
                        String(curr.name+"Count)");

                        Ln;
                        Tab;Tab;String('END;');Ln;

                        Tab;Tab;
                        String('FOR i');
                        Int(stackCount-1);
                        String(':=0 TO ');
                        Familia;
                        String(curr.name+"Count-1
DO");Ln;

                        CASE
curr.typ.BaseTyp.BaseTyp.form  OF

                        bool:
                                Tab;Tab;Tab;
                                String("rd.ReadBool(");
                                Familia;
                                String(curr.name+"[i");
                                Int(stackCount-1);
                                String("]);");Ln

                        |char:
                                Tab;Tab;Tab;
                                String("rd.ReadChar(");
                                Familia;
                                String(curr.name+"[i");
                                Int(stackCount-1);
                                String("]);");Ln
```

```
                                        |sInt:
                                                Tab;Tab;Tab;
                                                String("rd.ReadSInt(");
                                                Familia;
                                                String(curr.name+"[i");
                                                Int(stackCount-1);
                                                String("]);");Ln

                                        |int:
                                                Tab;Tab;Tab;
                                                String("rd.ReadInt(");
                                                Familia;
                                                String(curr.name+"[i");
                                                Int(stackCount-1);
                                                String("]);");Ln

                                        |sReal:
                                                Tab;Tab;Tab;
                                                String("rd.ReadSReal(");
                                                Familia;
                                                String(curr.name+"[i");
                                                Int(stackCount-1);
                                                String("]);");Ln

                                        |real:
                                                Tab;Tab;Tab;
                                                String("rd.ReadReal(");
                                                Familia;
                                                String(curr.name+"[i");
                                                Int(stackCount-1);
                                                String("]);");Ln

                                        |comp:
                                                con.Tab;con.String(" comp
");

                                                CASE
curr.typ.BaseTyp.BaseTyp.comp OF
                                                record:
RECORD "+curr.typ.BaseTyp.BaseTyp.strobj.name$+" *)");
                                                        Tab;Tab;String("(*
                                                        Ln;
        stackArrayFlag[stackCount]:=TRUE;
        ScanRecord(curr,curr.typ.BaseTyp.BaseTyp.link)
                                                ELSE
                                                END


                                        ELSE

                                        END;

                                        Tab;Tab;
                                        String('END;');Ln
                                END
                                |comp:
                                con.Tab;con.String(" comp ");

                                CASE curr.typ.comp OF
                                record:
                                        Tab;Tab;String("(* RECORD
"+curr.typ.strobj.name+" *)");
                                        Ln;
        stackArrayFlag[stackCount]:=FALSE;
                                        ScanRecord(curr,curr.typ.link);

                                ELSE
                                END
```

```
                                        ELSE
                                                con.Tab;con.String(" do define ")
                                        END;
                                        con.Ln;
                                        curr:=curr.link
                                END

                        END

                ELSE
                        con.Ln;con.String("ScanRecordUndefined "); con.Int(stackCount); con.Ln;
                        IF  stackMax<stackCount  THEN
                                stackMax:=stackCount
                        END;

                        IF  rec#NIL  THEN
                                curr:=rec;

                                WHILE  (curr#NIL)  DO
                                        con.Tab;con.String(curr.name$);
                                        CASE  curr.typ.form  OF
                                                bool:
                                                        con.Tab;con.String(" BOOLEAN ");
                                                |int:
                                                        con.Tab;con.String(" INTEGER ");
                                                |real:
                                                        con.Tab;con.String(" REAL ");
                                                |pointer:
                                                        con.Tab;con.String(" pointer ");

                                                        con.Ln;
                                                        ScanPointer(curr.typ.BaseTyp);

                                                |comp:
                                                        con.Tab;con.String(" comp ");

                                                        CASE curr.typ.comp OF
                                                        dynArray:
                                                                        con.Ln;
                                                                        ScanDynArray (curr);

                                                        | record:
                                                                con.Tab;con.String("RECORD ");

con.Tab;con.String(curr.typ.strobj.name$);

                                                                con.Ln;
                                                                ScanRecord(curr,curr.typ.link);

                                                        ELSE

                                                        END;

                                                |char:
                                                        con.Tab;con.String(" CHAR ");

                                        ELSE
                                                        con.Tab;con.String(" do define ");
                                        END;
                                        con.Ln;
                                        curr:=curr.link
                                END;

                        END;


        END;


        DEC(stackCount);

END ScanRecord;

PROCEDURE ScanDynArray (curr:DevCPT.Object);
```

```
BEGIN
        con.Tab;con.String("DYNARRAY "); con.Int(curr.typ.n); con.String(" ");
        con.String("OF ");


                con.String(curr.name$);


                        CASE  curr.typ.comp  OF
                        dynArray:
                                        con.Ln;
                                        ScanDynArray (curr.typ.BaseTyp.strobj);
                        | record:
                                con.Tab;con.String("RECORD ");

                                con.Ln;

                                ScanRecord(curr,curr.typ.link);


                        ELSE

                        END;




END ScanDynArray;


PROCEDURE ScanPointer (ptr:DevCPT.Struct);
        VAR
BEGIN

                con.Tab;con.String(" POINTER TO ");
                CASE  ptr.comp  OF
                dynArray:
                        ScanDynArray(ptr.BaseTyp.strobj)

                | record:
                        con.Tab;con.String("RECORD");
                        con.Tab;con.String(ptr.strobj.name$);

                        con.Ln;
                        ScanRecord(ptr.strobj,ptr.link);
                ELSE

                END;


END ScanPointer;

PROCEDURE MestnoeImya (obj: DevCPT.Object):POINTER TO ARRAY OF SHORTCHAR;
        VAR  p:POINTER TO ARRAY OF SHORTCHAR;
                        i:INTEGER;
BEGIN
        NEW(p,LEN(obj.name^));
        i:=0;
        WHILE  obj.name^[i]#0X  DO
                p[i]:=obj.name^[i];
                p[i+1]:=0X;
                INC(i);
        END;

        RETURN  p
END MestnoeImya;


PROCEDURE ProverkaKandidata (obj: DevCPT.Object);
        VAR  i,j:INTEGER;
                kandidat:POINTER TO ARRAY OF SHORTCHAR;
                suffix:ARRAY 5 OF SHORTCHAR;
```

```
BEGIN

        IF  (obj.typ.form= pointer)
                &(obj.typ.BaseTyp.form= comp)
                &(obj.typ.BaseTyp.attribute = extAttr ) THEN
                kandidat:=MestnoeImya(obj);

                Ln;
                String(" жазуу "+kandidat+" -маалымат сактагычка талапкер "); Ln;Ln;

                i:=0;
                WHILE  kandidat[i]#0X  DO
                        INC(i)
                END;

                IF  i>3  THEN
                        FOR  j :=0   TO 4   DO
                                suffix[j]:=kandidat[i-4+j];
                        END;

                        String(" теги "+suffix); Ln;Ln;

                        IF  suffix='Data'  THEN

                                String(" шартка туура келди "); Ln;Ln;
                                AppendUniqueObject(root,obj,rootCount);


                        ELSE
                                String(" шартка туура келбеди "); Ln;Ln;
                        END;
                ELSE
                        String(" шартка туура келбеди "); Ln;Ln;
                END;


        END;

END ProverkaKandidata;


PROCEDURE MyOut ();
VAR  i:INTEGER;
        curr:tObjectList;
BEGIN

        String("*)");Ln;

        String("TYPE");Ln;


        curr:=root;
        WHILE curr#NIL DO

                Tab;String(curr.val.name+"Robot* = POINTER TO RECORD ( "
                +OriginalModuleName+"."+curr.val.name+" );");
                 Ln;
                Tab;String("END;");Ln;Ln;
                curr:=curr.next
        END;

        curr:=root;
        WHILE curr#NIL DO

                mode:=Undefined;
                ScanRecord(curr.val,curr.val.right.typ.link);

                Tab; String(
                "PROCEDURE (m: "+curr.val.name+"Robot ) Memory2Binfile*(folder,file:ARRAY OF
CHAR),NEW;"); Ln;
                Tab; String("VAR i");
                FOR  i :=0   TO  stackMax-1  DO
                        String(",i");Int(i);
                END;
```

```
                        Tab; String(":INTEGER;");Ln;
                        Tab; String("wr: Stores.Writer;");Ln;
                        Tab; String("loc: Files.Locator;");Ln;
                        Tab; String("f: Files.File;");Ln;
                        Tab; String("res:INTEGER;");Ln;


                        Tab; String("BEGIN");Ln;
                        Tab;Tab; String("loc := Files.dir.This(folder$);");Ln;
                        Tab;Tab; String("f:= Files.dir.New( loc, Files.dontAsk ); ASSERT( f # NIL );");Ln;
                        Tab;Tab; String("wr.ConnectTo( f );");Ln;

                        mode:=Write; stackArrayFlag[stackCount]:=FALSE;
                        ScanRecord(curr.val,curr.val.right.typ.link);

                        Tab;Tab; String("wr.ConnectTo( NIL );");Ln;
                        Tab;Tab; String("f.Register(file$ ,'', Files.dontAsk, res );");Ln;
                        Tab;Tab; String("f.Close;");Ln;

                        Tab; String("END Memory2Binfile;");Ln;

                        Ln;


                        Tab; String(
                        "PROCEDURE (m: "+curr.val.name+"Robot ) Binfile2Memory*(folder,file:ARRAY OF
CHAR),NEW;");Ln;
                        Tab; String("VAR i");
                        FOR  i :=0   TO  stackMax-1  DO
                                String(",i");Int(i);
                        END;
                        Tab; String(":INTEGER;");Ln;
                        Tab; String("rd: Stores.Reader;");Ln;
                        Tab; String("loc: Files.Locator;");Ln;
                        Tab; String("f: Files.File;");Ln;
                        Tab; String("res:INTEGER;");Ln;

                        Tab; String("BEGIN");Ln;
                        Tab;Tab; String("loc := Files.dir.This(folder$);");Ln;
                        Tab;Tab; String("f:= Files.dir.Old( loc, file$, Files.shared );");Ln;
                        Tab;Tab; String("rd.ConnectTo( f );");Ln;

                        mode:=Read;stackArrayFlag[stackCount]:=FALSE;
                        ScanRecord(curr.val,curr.val.right.typ.link);

                        Tab;Tab; String("rd.ConnectTo( NIL );");Ln;
                        Tab;Tab; String("f.Close;");Ln;

                        Tab; String("END Binfile2Memory;");Ln;

                        Ln;

                        curr:=curr.next
                END;

        END MyOut;

        PROCEDURE IsHook(typ: DevCPT.Struct): BOOLEAN;
        BEGIN
                WHILE ((typ.form = pointer) OR (typ.form = comp)) & (typ.BaseTyp # NIL) DO typ :=
typ.BaseTyp END;
                RETURN (DevCPT.GlbMod[typ.mno].name^ = "Kernel") & (typ.strobj.name^ = "Hook^")
        END IsHook;

        (* auxilliary *)

        PROCEDURE GetSelection (VAR text: TextModels.Model; VAR beg, end: INTEGER);
                VAR c: TextControllers.Controller;
        BEGIN
                c := TextControllers.Focus();
                IF (c # NIL) & c.HasSelection() THEN
                        text := c.text; c.GetSelection(beg, end)
                ELSE
                        text := NIL
```

```
		END
	END GetSelection;

	PROCEDURE GetQualIdent (VAR mod, ident: DevCPT.Name; VAR t: TextModels.Model);
		VAR s: TextMappers.Scanner; beg, end: INTEGER;
	BEGIN
		GetSelection(t, beg, end);
		IF t # NIL THEN
			s.ConnectTo(t); s.SetOpts({7});	(* acceptUnderscores *)
			s.SetPos(beg); s.Scan;
			IF (s.type = TextMappers.string) & (s.len < LEN(mod)) THEN
				mod := SHORT(s.string$); s.Scan;
				IF (s.type = TextMappers.char) & (s.char = ".") & (s.Pos() <= end) THEN
					s.Scan;
					IF (s.type = TextMappers.string) & (s.len < LEN(ident)) THEN
						ident := SHORT(s.string$)
					ELSE mod[0] := 0X; ident[0] := 0X
					END
				ELSE ident[0] := 0X
				END
			ELSE mod[0] := 0X; ident[0] := 0X
			END
		ELSE mod[0] := 0X; ident[0] := 0X
		END
	END GetQualIdent;

	PROCEDURE Scan (VAR s: TextMappers.Scanner);
	BEGIN
		s.Scan;
		IF s.type = TextMappers.string THEN
			IF s.string = "IMPORT" THEN s.type := import
			ELSIF s.string = "MODULE" THEN s.type := module
			END
		ELSIF s.type = TextMappers.char THEN
			IF s.char = ";" THEN s.type := semicolon
			ELSIF s.char = ":" THEN
				IF s.rider.char = "=" THEN s.rider.Read; s.type := becomes END
			ELSIF s.char = "(" THEN
				IF s.rider.char = "*" THEN
					s.rider.Read;
					REPEAT Scan(s) UNTIL (s.type = TextMappers.eot) OR (s.type =
comEnd);
					Scan(s)
				END
			ELSIF s.char = "*" THEN
				IF s.rider.char = ")" THEN s.rider.Read; s.type := comEnd END
			END
		END
	END Scan;

	PROCEDURE CheckModName (VAR mod: DevCPT.Name; t: TextModels.Model);
		VAR s: TextMappers.Scanner;
	BEGIN
		s.ConnectTo(t); s.SetPos(0); Scan(s);
		IF s.type = module THEN
			Scan(s);
			WHILE (s.type # TextMappers.eot) & (s.type # import) DO Scan(s) END;
			WHILE (s.type # TextMappers.eot) & (s.type # semicolon)
					& ((s.type # TextMappers.string) OR (s.string # mod)) DO Scan(s)
END;
			IF s.type = TextMappers.string THEN
				Scan(s);
				IF s.type = becomes THEN
					Scan(s);
					IF s.type = TextMappers.string THEN
						mod := SHORT(s.string$)
					END
				END
			END
		END
	END CheckModName;


	PROCEDURE NewRuler (): TextRulers.Ruler;
```

```
		VAR ra: TextRulers.Attributes; p: TextRulers.Prop;
	BEGIN
		NEW(p); p.valid := {TextRulers.right, TextRulers.opts};
		p.opts.val := {(*TextRulers.rightFixed*)}; p.opts.mask := p.opts.val;
		p.right := width;
		NEW(ra); ra.InitFromProp(p);
		RETURN TextRulers.dir.New(TextRulers.dir.NewStyle(ra))
	END NewRuler;

	PROCEDURE Append (VAR d: ARRAY OF CHAR; s: ARRAY OF SHORTCHAR);
		VAR l, ld, ls: INTEGER;
	BEGIN
		l := LEN(d); ld := LEN(d$); ls := LEN(s$);
		IF ld + ls >= l THEN
			s[l - ld - 1] := 0X; d := d + s;
			d[l - 2] := "."; d[l - 3] := "."; d[l - 4] := "."
		ELSE
			d := d + s
		END
	END Append;

	PROCEDURE IsLegal (name: POINTER TO ARRAY OF SHORTCHAR): BOOLEAN;
		VAR i: SHORTINT;
	BEGIN
		IF name^ = "" THEN RETURN FALSE END;
		IF name[0] = "@" THEN RETURN global.hints END;
		i := 0;
		WHILE name[i] # 0X DO INC(i) END;
		RETURN name[i-1] # "^"
	END IsLegal;


	(* output formatting *)

	PROCEDURE Char (ch: SHORTCHAR);
	BEGIN
		global.out.WriteChar(ch)
	END Char;

	PROCEDURE String (s: ARRAY OF SHORTCHAR);
	BEGIN
		global.out.WriteSString(s)
	END String;

	PROCEDURE StringConst (s: ARRAY OF SHORTCHAR; long: BOOLEAN);
		VAR i, x, y: INTEGER; quoted, first: BOOLEAN;
	BEGIN
		IF long THEN
			i := 0; REPEAT DevCPM.GetUtf8(s, y, i) UNTIL (y = 0) OR (y >= 100H);
			IF y = 0 THEN global.out.WriteString("LONG(") END
		END;
		i := 0; quoted := FALSE; first := TRUE;
		IF long THEN DevCPM.GetUtf8(s, x, i) ELSE x := ORD(s[i]); INC(i) END;
		WHILE x # 0 DO
			IF (x < ORD(" ")) OR (x >= 100H) THEN
				IF quoted THEN global.out.WriteChar("'") END;
				IF ~first THEN global.out.WriteString(" + ") END;
				IF x >= 0A00H THEN global.out.WriteIntForm(x, TextMappers.hexadecimal, 4,
"0", FALSE)
				ELSIF x >= 0A0H THEN global.out.WriteIntForm(x, TextMappers.hexadecimal,
3, "0", FALSE)
				ELSE global.out.WriteIntForm(x, TextMappers.hexadecimal, 2, "0", FALSE)
				END;
				global.out.WriteChar("X");
				quoted := FALSE
			ELSE
				IF ~quoted THEN
					IF ~first THEN global.out.WriteString(" + ") END;
					global.out.WriteChar("'")
				END;
				global.out.WriteChar(CHR(x));
				quoted := TRUE
			END;
			IF long THEN DevCPM.GetUtf8(s, x, i) ELSE x := ORD(s[i]); INC(i) END;
```

```
                        first := FALSE
                END;
                IF first THEN global.out.WriteString('"') END;
                IF quoted THEN global.out.WriteChar('"') END;
                IF long & (y = 0) THEN global.out.WriteChar(")") END
        END StringConst;

        PROCEDURE ProperString (s: ARRAY OF SHORTCHAR);
                VAR i: SHORTINT;
        BEGIN
                IF s # "" THEN
                        i := 0; WHILE (s[i] # 0X) & (s[i] # "^") DO global.out.WriteChar(s[i]); INC(i) END
                END
        END ProperString;

        PROCEDURE Keyword (s: ARRAY OF SHORTCHAR);
                VAR a: TextModels.Attributes;
        BEGIN
                IF global.keyAttr # NIL THEN a := global.out.rider.attr; global.out.rider.SetAttr(global.keyAttr)
END;
                global.out.WriteSString(s);
                IF global.keyAttr # NIL THEN global.out.rider.SetAttr(a) END
        END Keyword;

        PROCEDURE Section (VAR out: TextMappers.Formatter; s: ARRAY OF SHORTCHAR);
                VAR a: TextModels.Attributes;
        BEGIN
                IF global.sectAttr # NIL THEN a := out.rider.attr; out.rider.SetAttr(global.sectAttr) END;
                out.WriteSString(s);
                IF global.sectAttr # NIL THEN out.rider.SetAttr(a) END
        END Section;

        PROCEDURE Int (i: INTEGER);
        BEGIN
                global.out.WriteInt(i)
        END Int;

        PROCEDURE Hex (x, n: INTEGER);
        BEGIN
                IF n > 1 THEN Hex(x DIV 16, n - 1) END;
                x := x MOD 16;
                IF x >= 10 THEN global.out.WriteChar(SHORT(CHR(x + ORD("A") - 10)))
                ELSE global.out.WriteChar(SHORT(CHR(x + ORD("0"))))
                END
        END Hex;

        PROCEDURE Tab ();
        BEGIN
                global.out.WriteTab;
        END Tab;

        PROCEDURE Ln;
        BEGIN
                global.out.WriteLn
        END Ln;

        PROCEDURE Indent;
                VAR i: SHORTINT;
        BEGIN
                IF global.gap THEN global.gap := FALSE; Ln END;
                i := global.level; WHILE i > 0 DO global.out.WriteTab; DEC(i) END
        END Indent;

        PROCEDURE Guid (ext: DevCPT.ConstExt);
        BEGIN
                Char("{");
                Hex(ORD(ext[2]) + 256 * ORD(ext[3]), 4);
                Hex(ORD(ext[0]) + 256 * ORD(ext[1]), 4);
                Char("-");
                Hex(ORD(ext[4]) + 256 * ORD(ext[5]), 4);
                Char("-");
                Hex(ORD(ext[6]) + 256 * ORD(ext[7]), 4);
                Char("-");
                Hex(ORD(ext[8]), 2);
```

```
		Hex(ORD(ext[9]), 2);
		Char("-");
		Hex(ORD(ext[10]), 2);
		Hex(ORD(ext[11]), 2);
		Hex(ORD(ext[12]), 2);
		Hex(ORD(ext[13]), 2);
		Hex(ORD(ext[14]), 2);
		Hex(ORD(ext[15]), 2);
		Char("}")
	END Guid;


	(* special marks *)

	PROCEDURE Hint (label: ARRAY OF SHORTCHAR; i: INTEGER; adj: BOOLEAN);
	BEGIN
		IF global.hints THEN
			Char("["); String(label);
			IF adj & (0 <= i) & (i < 10) THEN Char(TextModels.digitspace) END;
			Int(i);
			String("] ")
		END
	END Hint;

	PROCEDURE Hint3 (label1, label2, label3: ARRAY OF SHORTCHAR; i1, i2, i3: INTEGER);
	BEGIN
		IF global.hints THEN
			Char("["); String(label1); Int(i1); String(label2); Int(i2); String(label3); Int(i3);
String("] ")
		END
	END Hint3;

	PROCEDURE Vis (i: INTEGER);
	BEGIN
		IF i # external THEN Char("-") END
	END Vis;

	PROCEDURE ProcSysFlag (flag: SHORTINT);
	BEGIN
		IF flag # 0 THEN
			String(" [");
			IF flag = -10 THEN String("ccall")
			ELSE Int(flag)
			END;
			Char("]")
		END
	END ProcSysFlag;

	PROCEDURE ParSysFlag (flag: SHORTINT);
		VAR pos: INTEGER;
	BEGIN
		IF flag # 0 THEN
			String(" [");
			pos := global.out.Pos();
			IF ODD(flag) THEN String("nil") END;
(*
			IF ODD(flag DIV 2) & (flag # 18) THEN
				IF global.out.Pos() # pos THEN String(", ") END;
				String("in")
			END;
			IF ODD(flag DIV 4) & (flag # 12) THEN
				IF global.out.Pos() # pos THEN String(", ") END;
				String("out")
			END;
*)
			IF flag DIV 8 = 1 THEN
				IF global.out.Pos() # pos THEN String(", ") END;
				String("new")
			ELSIF flag DIV 8 = 2 THEN
				IF global.out.Pos() # pos THEN String(", ") END;
				String("iid")
			ELSIF flag DIV 8 # 0 THEN
				IF global.out.Pos() # pos THEN String(", ") END;
				Int(flag DIV 8 * 8)
```

```
                        END;
                        Char("]")
                END
        END ParSysFlag;

        PROCEDURE StructSysFlag (typ: DevCPT.Struct);
                VAR flag: SHORTINT;
        BEGIN
                flag := typ.sysflag;
                IF (flag # 0) & ((typ.form # pointer) OR (flag = 2)) THEN
                        String(" [");
                        IF flag = 1 THEN String("untagged")
                        ELSIF flag = 2 THEN String("handle")
                        ELSIF flag = 3 THEN String("noalign")
                        ELSIF flag = 4 THEN String("align2")
                        ELSIF flag = 5 THEN String("align4")
                        ELSIF flag = 6 THEN String("align8")
                        ELSIF flag = 7 THEN String("union")
                        ELSIF flag = 10 THEN
                                IF typ.ext # NIL THEN Char('"'); String(typ.ext^); Char('"')
                                ELSE String("interface")
                                END
                        ELSIF flag = 20 THEN String("som")
                        ELSE Int(flag)
                        END;
                        Char("]")
                END
        END StructSysFlag;

        PROCEDURE SysStrings (obj: DevCPT.Object);
        BEGIN
                IF global.hints & ((obj.entry # NIL) OR (obj.library # NIL)) THEN
                        String(' ["');
                        IF obj.library # NIL THEN String(obj.library^); String('", "') END;
                        IF obj.entry # NIL THEN String(obj.entry^) END;
                        String('"]')
                END
        END SysStrings;


        (* non-terminals *)

        PROCEDURE ^ Structure (typ: DevCPT.Struct);

        PROCEDURE Qualifier (mno: SHORTINT);
        BEGIN
                IF (mno > 1) OR (mno = 1) & global.singleton THEN
                        global.modUsed[mno] := TRUE;
                        String(DevCPT.GlbMod[mno].name^); Char(".")
                END
        END Qualifier;

        PROCEDURE LocalName (obj: DevCPT.Object);
        BEGIN
                Qualifier(0); String(obj.name^)
        END LocalName;

        PROCEDURE TypName (typ: DevCPT.Struct; force: BOOLEAN);
        BEGIN
                IF force OR IsLegal(typ.strobj.name) THEN
                        IF (typ.mno > 1) OR (typ.mno = 1)
                                & ((typ.form >= pointer) OR (typ.BaseTyp # DevCPT.undftyp)) &
global.singleton
                        THEN
                                Qualifier(typ.mno)
                        ELSIF typ = DevCPT.sysptrtyp THEN          (* PTR is in SYSTEM *)
                                String("SYSTEM.");
                                global.systemUsed := TRUE
                        ELSIF
                                (typ = DevCPT.guidtyp) OR (typ = DevCPT.restyp) OR (typ = DevCPT.iunktyp)
OR (typ = DevCPT.punktyp)
                        THEN
                                String("COM.");
                                global.comUsed := TRUE
```

```
                        END;
                        IF force THEN ProperString(typ.strobj.name^) ELSE String(typ.strobj.name^) END
                ELSE
                        Structure(typ);
                END
        END TypName;

        PROCEDURE ParList (VAR par: DevCPT.Object);
        BEGIN
                IF par.mode = varPar THEN
                        IF par.vis = inPar THEN Keyword("IN")
                        ELSIF par.vis = outPar THEN Keyword("OUT")
                        ELSE Keyword("VAR")
                        END;
                        ParSysFlag(par.sysflag); Char(" ")
                END;
                String(par.name^);
                WHILE (par.link # NIL) & (par.link.typ = par.typ) & (par.link.mode = par.mode)
                                & (par.link.vis = par.vis) & (par.link.sysflag = par.sysflag) DO
                        par := par.link; String(", "); String(par.name^)
                END;
                String(": "); TypName(par.typ, FALSE)
        END ParList;

        PROCEDURE Signature (result: DevCPT.Struct; par: DevCPT.Object);
                VAR paren, res: BOOLEAN;
        BEGIN
                res := result # DevCPT.notyp; paren := res OR (par # NIL);
                IF paren THEN String(" (") END;
                IF par # NIL THEN
                        ParList(par); par := par.link;
                        WHILE par # NIL DO String("; "); ParList(par); par := par.link END
                END;
                IF paren THEN Char(")") END;
                IF res THEN String(": "); TypName(result, FALSE) END
        END Signature;

        PROCEDURE TProcs (rec: DevCPT.Struct; fld: DevCPT.Object; oldProcs: TProcList; VAR newProcs:
TProcList);
                VAR old: DevCPT.Object; p, elem: TProcList;
        BEGIN
                IF fld # NIL THEN
                        TProcs(rec, fld.left, oldProcs, newProcs);
                        IF (fld.mode = tProc) & IsLegal(fld.name) THEN
                                DevCPT.FindBaseField(fld.name^, rec, old);
                                IF (old = NIL) OR (fld.typ # old.typ) OR (* (rec.attribute # 0) & *)
(fld.conval.setval # old.conval.setval)
                                THEN
                                        IF global.extensioninterface OR global.clientinterface THEN
                                                (* do not show methods with equal name *)
                                                p := oldProcs;
                                                WHILE (p # NIL) & (p.fld.name^ # fld.name^) DO p := p.next
END
                                        END;
                                        IF p = NIL THEN
                                                NEW(elem); elem.next := newProcs; newProcs := elem;
                                                elem.fld := fld;
                                                IF old = NIL THEN INCL(elem.attr, newAttr) END;
                                                IF absAttr IN fld.conval.setval THEN INCL(elem.attr, absAttr)
                                                ELSIF empAttr IN fld.conval.setval THEN INCL(elem.attr,
empAttr)
                                                ELSIF extAttr IN fld.conval.setval THEN INCL(elem.attr,
extAttr)
                                                END
                                        END
                                END
                        END;
                        TProcs(rec, fld.right, oldProcs, newProcs)
                END
        END TProcs;

(*
        PROCEDURE AdjustTProcs (typ: DevCPT.Struct; fld: DevCPT.Object);
                VAR receiver, old: DevCPT.Object; base: DevCPT.Struct;
```

```
	BEGIN
		IF fld # NIL THEN
			AdjustTProcs(typ, fld.left);
			IF (fld.mode = tProc) & IsLegal(fld.name) THEN
				DevCPT.FindBaseField(fld.name^, typ, old);
				IF old # NIL THEN
					IF fld.conval.setval * {absAttr, empAttr, extAttr} = {} THEN
						old.conval.setval := old.conval.setval - {absAttr, empAttr,
extAttr}
					ELSIF (extAttr IN fld.conval.setval) & (empAttr IN old.conval.setval)
THEN
						(* do not list methods which only change attribute from empty
to extensible *)
						old.conval.setval := old.conval.setval + {extAttr} - {empAttr}
					END;
					IF fld.typ # old.typ THEN (* do not show base methods of covariant
overwritten methods *)
						old.typ := fld.typ; old.conval.setval := {}
					END;
				END
			END;
			AdjustTProcs(typ, fld.right)
		END;
		IF (typ.BaseTyp # NIL) & (typ.BaseTyp # DevCPT.iunktyp) THEN
			AdjustTProcs(typ.BaseTyp, typ.BaseTyp.link)
		END
	END AdjustTProcs;
*)

	PROCEDURE TypeboundProcs (typ: DevCPT.Struct; VAR cont: BOOLEAN; top: BOOLEAN;
		VAR list:
TProcList; recAttr: INTEGER);
		VAR receiver: DevCPT.Object; new, p, q: TProcList;
	BEGIN
		IF (typ # NIL) & (typ # DevCPT.iunktyp) THEN
			TProcs(typ, typ.link, list, new);

			p := list;
			WHILE new # NIL DO
				q := new.next; new.next := p; p := new; new := q
			END;
			list := p;

			IF global.flatten THEN
				TypeboundProcs(typ.BaseTyp, cont, FALSE, list, recAttr);
				IF (typ.BaseTyp = NIL) OR (typ.BaseTyp = DevCPT.iunktyp) THEN
(*
					IF global.extensioninterface & (recAttr IN {absAttr, extAttr}) THEN
						IF cont THEN Char(";") END;
						Ln; Indent;
						String("(a: ANYPTR) FINALIZE-, "); Keyword("NEW"); String(",
"); Keyword("EMPTY");
						cont := TRUE
					END
*)
				END
			END;
			IF top THEN (* writeList *)
				WHILE list # NIL DO
					IF ~global.clientinterface & ~global.extensioninterface (* default *)
					OR global.clientinterface & (list.fld.vis = external)
					OR global.extensioninterface & (recAttr IN {absAttr, extAttr}) &
(list.attr * {absAttr, empAttr, extAttr} # {})
					THEN

						IF cont THEN Char(";") END;
						Ln;
						receiver := list.fld.link;
						Indent; Hint("entry ", list.fld.num, TRUE);
	(*
						Keyword("PROCEDURE"); String(" (");
	*)
						Char("(");
						IF receiver.mode = varPar THEN
```

```
                                                IF receiver.vis = inPar THEN Keyword("IN") ELSE
Keyword("VAR") END;
                                                Char(" ")
                                        END;
                                        String(receiver.name^); String(": ");
                                        IF IsLegal(receiver.typ.strobj.name) & (receiver.typ.mno = 1)
THEN
                                                String(receiver.typ.strobj.name^)
                                        ELSE TypName(receiver.typ, FALSE)
                                        END;
                                        String(") "); String(list.fld.name^); Vis(list.fld.vis);
                                        SysStrings(list.fld);
                                        Signature(list.fld.typ, receiver.link);
                                        IF newAttr IN list.attr THEN String(", "); Keyword("NEW")
END;
                                        IF absAttr IN list.attr THEN String(", "); Keyword("ABSTRACT")
                                        ELSIF empAttr IN list.attr THEN String(", ");
Keyword("EMPTY")
                                        ELSIF extAttr IN list.attr THEN String(", ");
Keyword("EXTENSIBLE")
                                        END;
                                        cont := TRUE
                                END;
                                list := list.next
                        END
                END
        END
END TypeboundProcs;

PROCEDURE Flds (typ: DevCPT.Struct; VAR cont: BOOLEAN);
        VAR fld: DevCPT.Object;
BEGIN
        fld := typ.link;
        WHILE (fld # NIL) & (fld.mode = field) DO
                IF IsLegal(fld.name) THEN
                        IF cont THEN Char(";") END;
                        Ln;
                        Indent; Hint("offset ", fld.adr, TRUE);
                        IF typ.mno > 1 THEN String("(* "); TypName(typ, TRUE); String(" *) ") END;
                        String(fld.name^);
                        WHILE (fld.link # NIL) & (fld.link.typ = fld.typ) & (fld.link.name # NIL) DO
                                Vis(fld.vis); fld := fld.link; String(", "); String(fld.name^)
                        END;
                        Vis(fld.vis); String(": "); TypName(fld.typ, FALSE);
                        cont := TRUE
                END;
                fld := fld.link
        END
END Flds;

PROCEDURE Fields (typ: DevCPT.Struct; VAR cont: BOOLEAN);
BEGIN
        IF typ # NIL THEN
                IF global.flatten THEN Fields(typ.BaseTyp, cont) END;
                Flds(typ, cont)
        END
END Fields;

PROCEDURE BaseTypes (typ: DevCPT.Struct);
BEGIN
        IF typ.BaseTyp # NIL THEN
                Char("("); TypName(typ.BaseTyp, TRUE);
                IF global.flatten THEN BaseTypes(typ.BaseTyp) END;
                Char(")")
        END
END BaseTypes;

PROCEDURE Structure (typ: DevCPT.Struct);
        VAR cont: BOOLEAN; p: TProcList;
BEGIN
        INC(global.level);
        CASE typ.form OF
          pointer:
                Keyword("POINTER"); StructSysFlag(typ); Char(" ");
```

```
                        Keyword("TO"); Char(" "); DEC(global.level); TypName(typ.BaseTyp, FALSE);
INC(global.level)
                | procTyp:
                        Keyword("PROCEDURE"); Signature(typ.BaseTyp, typ.link)
                | comp:
                        CASE typ.comp OF
                          array:
                                Keyword("ARRAY"); StructSysFlag(typ); Char(" "); Int(typ.n); Char(" ");
                                Keyword("OF"); Char(" "); TypName(typ.BaseTyp, FALSE)
                        | dynArray:
                                Keyword("ARRAY"); StructSysFlag(typ); Char(" ");
                                Keyword("OF"); Char(" "); TypName(typ.BaseTyp, FALSE)
                        | record:
                                IF typ.attribute = absAttr THEN Keyword("ABSTRACT ")
                                ELSIF typ.attribute = limAttr THEN Keyword("LIMITED ")
                                ELSIF typ.attribute = extAttr THEN Keyword("EXTENSIBLE ");
                                END;
                                Keyword("RECORD"); StructSysFlag(typ);
                                Char(" "); Hint3("tprocs ", ", size ", ", align ", typ.n, typ.size, typ.align);
                                cont := FALSE;
                                BaseTypes(typ); Fields(typ, cont); TypeboundProcs(typ, cont, TRUE, p,
typ.attribute);
                                IF cont THEN Ln; DEC(global.level); Indent; INC(global.level) ELSE Char (" ")
END;
                                Keyword("END")
                        END
                ELSE
                        IF (typ.BaseTyp # DevCPT.undftyp) THEN TypName(typ.BaseTyp, FALSE) END(* alias
structures *)
                END;
                DEC(global.level);

        END Structure;

        PROCEDURE Const (obj: DevCPT.Object);
                VAR con: DevCPT.Const; s: SET; i: SHORTINT; x, y: INTEGER; r: REAL;
        BEGIN
                Indent; LocalName(obj); SysStrings(obj); String(" = ");
                con := obj.conval;
                CASE obj.typ.form OF
                  bool:
                        IF con.intval = 1 THEN String("TRUE") ELSE String("FALSE") END
                | sChar, char:
                        IF con.intval >= 0A000H THEN
                                Hex(con.intval, 5); Char("X")
                        ELSIF con.intval >= 0100H THEN
                                Hex(con.intval, 4); Char("X")
                        ELSIF con.intval >= 0A0H THEN
                                Hex(con.intval, 3); Char("X")
                        ELSIF (con.intval >= 32) & (con.intval <= 126) THEN
                                Char(22X); Char(SHORT(CHR(con.intval))); Char(22X)
                        ELSE
                                Hex(con.intval, 2); Char("X")
                        END
                | byte, sInt, int:
                        Int(con.intval)
                | lInt:
                        y := SHORT(ENTIER((con.realval + con.intval) / 4294967296.0));
                        r := con.realval + con.intval - y * 4294967296.0;
                        IF r > MAX(INTEGER) THEN r := r - 4294967296.0 END;
                        x := SHORT(ENTIER(r));
                        Hex(y, 8); Hex(x, 8); Char("L")
                | set:
                        Char("{"); i := 0; s := con.setval;
                        WHILE i <= MAX(SET) DO
                                IF i IN s THEN Int(i); EXCL(s, i);
                                        IF s # {} THEN String(", ") END
                                END;
                                INC(i)
                        END;
                        Char("}")
                | sReal, real:
                        global.out.WriteReal(con.realval)
                | sString:
```

```
                        StringConst(con.ext^, FALSE)
                | string:
                        StringConst(con.ext^, TRUE)
                | nilTyp:
                        Keyword("NIL")
                | comp: (* guid *)
                        Char("{"); Guid(con.ext); Char("}")
                END;
                Char(";"); Ln
        END Const;

        PROCEDURE AliasType (typ: DevCPT.Struct);
        BEGIN
                IF global.singleton & IsLegal(typ.strobj.name) THEN
                        WHILE typ.BaseTyp # DevCPT.undftyp DO
                                Char(";"); Ln; Indent;
                                String(typ.strobj.name^); SysStrings(typ.strobj); String(" = ");
                                IF typ.form >= pointer THEN

                                        Structure(typ); typ := DevCPT.int16typ
                                ELSE
                                        typ := typ.BaseTyp;
                                        TypName(typ, FALSE)
                                END
                        END
                END
        END AliasType;

        PROCEDURE NotExtRec(typ: DevCPT.Struct): BOOLEAN;
        BEGIN
                RETURN (typ.form = comp) & ((typ.comp # record) OR ~(typ.attribute IN {absAttr, extAttr}))
                        OR (typ.form = procTyp)
        END NotExtRec;

        PROCEDURE Type (obj: DevCPT.Object);
        BEGIN
                IF global.hideHooks & IsHook(obj.typ) THEN RETURN END;

                IF global.extensioninterface
(*
                & (obj.typ.strobj = obj) & ~((obj.typ.form < pointer) & (obj.typ.BaseTyp # DevCPT.undftyp))
*)
                & (NotExtRec(obj.typ)
                        OR
                        (obj.typ.form = pointer) & ~IsLegal(obj.typ.BaseTyp.strobj.name) &
                                NotExtRec(obj.typ.BaseTyp)
                )

                THEN
                        IF global.singleton THEN RETURN END;
                        global.out.rider.SetAttr(TextModels.NewColor(global.out.rider.attr, Ports.grey50));
                        IF global.pos < global.out.rider.Pos() THEN
                                global.out.rider.Base().SetAttr(global.pos, global.out.rider.Pos()-1,
global.out.rider.attr)
                        END
                END;

                ProverkaKandidata (obj);

                Indent; LocalName(obj); SysStrings(obj); String(" = ");
                IF obj.typ.strobj # obj THEN
                        TypName(obj.typ, FALSE);
                        AliasType(obj.typ)
                ELSIF (obj.typ.form < pointer) & (obj.typ.BaseTyp # DevCPT.undftyp) THEN
                        TypName(obj.typ.BaseTyp, FALSE);
                        AliasType(obj.typ.BaseTyp)
                ELSE


                Structure(obj.typ); (*она рекурсивная*)


                END;
                Char(";"); Ln;
```

```
		global.gap := TRUE
	END Type;

	PROCEDURE PtrType (obj: DevCPT.Object);
		VAR base: DevCPT.Object;
	BEGIN
		Type(obj);
		base := obj.typ.BaseTyp.strobj;
		IF (base # NIL) & (base # DevCPT.anytyp.strobj) & (base # DevCPT.iunktyp.strobj)
				& (base.vis # internal) & (base.typ.strobj # NIL) & ~(global.hideHooks &
IsHook(base.typ)) THEN
			(* show named record with pointer; avoid repetition *)

		IF global.extensioninterface
		& (base.typ.strobj = base) & ~((obj.typ.form < pointer) & (obj.typ.BaseTyp # DevCPT.undftyp))
		& (NotExtRec(base.typ)
			OR
			(base.typ.form = pointer) & ~IsLegal(base.typ.BaseTyp.strobj.name) &
				NotExtRec(base.typ.BaseTyp)
		)

		THEN
			IF global.singleton THEN
				IF global.pos < global.out.rider.Pos() THEN
					global.out.rider.Base().Delete(global.pos, global.out.rider.Pos());
					global.out.rider.SetPos(global.pos)
				END;
				RETURN
			END;
			global.out.rider.SetAttr(TextModels.NewColor(global.out.rider.attr, Ports.grey50));
			IF global.pos < global.out.rider.Pos() THEN
				global.out.rider.Base().SetAttr(global.pos, global.out.rider.Pos()-1,
global.out.rider.attr)
			END
		END;

			global.gap := FALSE;
			Indent;
			String(base.name^); SysStrings(base); String(" = ");
			IF base.typ.strobj # base THEN
				TypName(base.typ, FALSE);
				AliasType(obj.typ)
			ELSIF (obj.typ.form < pointer) & (obj.typ.BaseTyp # DevCPT.undftyp) THEN
				TypName(obj.typ.BaseTyp, FALSE);
				AliasType(obj.typ.BaseTyp)
			ELSE

				Structure(base.typ)
			END;
			Char(";"); Ln;
			base.vis := internal;
			global.gap := TRUE
		END
	END PtrType;

(*

	PROCEDURE Var (obj: DevCPT.Object);
	BEGIN
		IF global.hideHooks & IsHook(obj.typ) THEN RETURN END;

		String('root record name := '+obj.name);
		Tab;

		здесь искать суффикс

		Indent; LocalName(obj); Vis(obj.vis);

		SysStrings(obj); String(": ");
		TypName(obj.typ, FALSE); Char(";");

		Ln
	END Var;
	*)
```

```
	(* tree traversal *)

	PROCEDURE Objects (obj: DevCPT.Object; mode: SET);
		VAR m: BYTE; a: TextModels.Attributes;
	BEGIN
		IF obj # NIL THEN
			INC(lev); INC(num); max := MAX(max, lev);
			Objects(obj.left, mode);
			m := obj.mode; IF (m = type) & (obj.typ.form = pointer) THEN m := ptrType END;
			IF (m IN mode) & (obj.vis # internal) & (obj.name # NIL) &
					(~global.singleton OR (obj.name^ = global.thisObj)) THEN
				global.pos := global.out.rider.Pos(); a := global.out.rider.attr;
				CASE m OF
				  const: Const(obj)
				| type: Type(obj)
				| ptrType: PtrType(obj)
				| var: (* Var(obj) *)
				| xProc, cProc, iProc:
				END;
				global.out.rider.SetAttr(a)
			END;
			Objects(obj.right, mode);
			DEC(lev)
		END
	END Objects;


	(* definition formatting *)

	PROCEDURE PutSection (VAR out: TextMappers.Formatter; s: ARRAY OF SHORTCHAR);
		VAR buf, def: TextModels.Model; i: INTEGER;
	BEGIN
		buf := global.out.rider.Base();
		IF buf.Length() > 0 THEN
			IF ~global.singleton THEN Ln END;
			def := out.rider.Base(); out.SetPos(def.Length());
			IF s # "" THEN    (* prepend section keyword *)
				i := global.level - 1; WHILE i > 0 DO out.WriteTab; DEC(i) END;
				Section(out, s); out.WriteLn
			END;
			def.Append(buf);
			global.pos := 0;
			global.out.rider.SetPos(0)
		END;
		global.gap := FALSE
	END PutSection;

	PROCEDURE Scope (def: TextModels.Model; this: DevCPT.Name);
		VAR out: TextMappers.Formatter; i: SHORTINT; a: TextModels.Attributes;
	BEGIN
		global.gap := FALSE; global.thisObj := this; global.singleton := (this # "");
		global.systemUsed := FALSE; global.comUsed := FALSE;
		i := 1; WHILE i < LEN(global.modUsed) DO global.modUsed[i] := FALSE; INC(i) END;
		out.ConnectTo(def);
		INC(global.level);
		IF ~(global.extensioninterface & global.singleton) THEN
			IF global.extensioninterface THEN
				a := global.out.rider.attr; global.out.rider.SetAttr(TextModels.NewColor(a,
Ports.grey50))
			END;
			Objects(DevCPT.GlbMod[1].right, {const});
			IF global.extensioninterface THEN global.out.rider.SetAttr(a) END;
			PutSection(out, "CONST")
		END;
		Objects(DevCPT.GlbMod[1].right, {ptrType});
		Objects(DevCPT.GlbMod[1].right, {type}); PutSection(out, "TYPE");
		IF ~global.extensioninterface THEN
			Objects(DevCPT.GlbMod[1].right, {var}); PutSection(out, "VAR");
		END;
		DEC(global.level);
		num := 0; lev := 0; max := 0;
		Objects(DevCPT.GlbMod[1].right, {xProc, cProc}); PutSection(out, "")
```

```
	END Scope;

	PROCEDURE Modules;
		VAR i, j, n: SHORTINT;
	BEGIN
		n := 0; j := 2;
		IF global.systemUsed THEN INC(n) END;
		IF global.comUsed THEN INC(n) END;
		WHILE j < DevCPT.nofGmod DO
			IF global.modUsed[j] THEN INC(n) END;
			INC(j)
		END;

		String("IMPORT Files,Stores,"+OriginalModuleName);

		IF n > 0 THEN
(*			Indent; Section(global.out, "IMPORT"); *)
			Tab;
			String(",");
			INC(global.level);
			IF n < 10 THEN Char(" ") ELSE Ln; Indent END;
			i := 0; j := 2;
			IF global.systemUsed THEN
				String("SYSTEM"); INC(i);
				IF i < n THEN String(", ") END
			END;
			IF global.comUsed THEN
				String("COM"); INC(i);
				IF i < n THEN String(", ") END
			END;
			WHILE i < n DO
				IF global.modUsed[j] THEN
					String(DevCPT.GlbMod[j].name^); INC(i);
(*
					IF (DevCPT.GlbMod[j].strings # NIL) & (DevCPT.GlbMod[j].strings.ext
# NIL) THEN
						String(' ["'); String(DevCPT.GlbMod[j].strings.ext^);
String('"]')
					END;
*)
					IF i < n THEN
						Char(",");
						IF i MOD 10 = 0 THEN Ln; Indent ELSE Char(" ") END
					END
				END;
				INC(j)
			END;
		END;

		Char(";"); Ln;
		String("(*");Ln;
		Ln;

	END Modules;


	(* compiler interfacing *)

	PROCEDURE OpenCompiler (mod: DevCPT.Name; VAR noerr: BOOLEAN);
		VAR null: TextModels.Model; main: DevCPT.Name;
	BEGIN
		null := TextModels.dir.New();
		DevCPM.Init(null.NewReader(NIL), null);
(*
		DevCPT.Init({}); DevCPT.Open(mod); DevCPT.SelfName := "AvoidErr154";
*)
		DevCPT.Init({}); main := "@mainMod"; DevCPT.Open(main);
		DevCPT.processor := 0;   (* accept all sym files *)
		DevCPT.Import("@notself", mod, noerr);
		IF ~DevCPM.noerr THEN noerr := FALSE END
	END OpenCompiler;

	PROCEDURE CloseCompiler;
	BEGIN
```

```
		DevCPT.Close;
		DevCPM.Close
	END CloseCompiler;


	PROCEDURE Browse (
		ident: DevCPT.Name; opts: ARRAY OF CHAR; VAR view: Views.View; VAR title: Views.Title
	);
		VAR def, buf: TextModels.Model; v: TextViews.View; h: BOOLEAN;
			p: Properties.BoundsPref; mod: DevCPT.Name; i: SHORTINT; str, str1: ARRAY 256 OF
CHAR;
	BEGIN
		mod := DevCPT.GlbMod[1].name^$;
		i := 0;
		global.hints := FALSE; global.flatten := FALSE; global.hideHooks := TRUE;
		global.clientinterface := FALSE; global.extensioninterface := FALSE;
		global.keyAttr := NIL; global.sectAttr := NIL;

		IF opts[0] = '@' THEN
			IF options.hints THEN opts := opts + "+" END;
			IF options.flatten THEN opts := opts + "!" END;
			IF options.formatted THEN opts := opts + "/" END
		END;

		WHILE opts[i] # 0X DO
			CASE opts[i] OF
			| '+': global.hints := TRUE
			| '!': global.flatten := TRUE
			| '&': global.hideHooks := FALSE
			| '/':
				(* global.keyAttr := TextModels.NewStyle(TextModels.dir.attr,
{Fonts.underline}); *)
				global.keyAttr := TextModels.NewWeight(TextModels.dir.attr, Fonts.bold);
				global.sectAttr := TextModels.NewWeight(TextModels.dir.attr, Fonts.bold)
			| 'c': global.clientinterface := TRUE
			| 'e' : global.extensioninterface := TRUE
			ELSE
			END;
			INC(i)
		END;
		IF global.clientinterface & global.extensioninterface THEN
			global.clientinterface := FALSE; global.extensioninterface := FALSE
		END;
		IF global.extensioninterface THEN global.hideHooks := FALSE END;
		def := TextModels.dir.New();
		buf := TextModels.dir.New(); global.out.ConnectTo(buf);
		IF ident # "" THEN
			h := global.hideHooks; global.hideHooks := FALSE;
			global.level := 0; Scope(def, ident);
			global.hideHooks := h;
			def.Append(buf);
			IF def.Length() > 0 THEN
				v := TextViews.dir.New(def);
				v.SetDefaults(NewRuler(), TextViews.dir.defAttr);
				p.w := Views.undefined; p.h := Views.undefined; Views.HandlePropMsg(v, p);
				title := mod$; Append(title, "."); Append(title, ident);
				view := Documents.dir.New(v, p.w, p.h)
			ELSE str := mod$; str1 := ident$; title := "";
				IF global.extensioninterface THEN
					Dialog.ShowParamMsg(noSuchExtItemExportedKey, str, str1, "")
				ELSE
					Dialog.ShowParamMsg(noSuchItemExportedKey, str, str1, "")
				END
			END
		ELSE
			global.level := 1; Scope(def, "");

			OriginalModuleName:=mod$;

			MyOut;

			Section(global.out, "END"); Char(" "); String(mod);
			Section(global.out, "Robot");
```

```
                        Char("."); Ln;
                        def.Append(buf);
                        Section(global.out, "MODULE"); Char(" "); String(mod);
                        Section(global.out, "Robot");
                        Char(";"); Ln; Ln;

                        Modules;
                        buf.Append(def);
                        v := TextViews.dir.New(buf);
                        v.SetDefaults(NewRuler(), TextViews.dir.defAttr);
                        title := mod$;
                        view := Documents.dir.New(v, width, height)
                END;
                global.out.ConnectTo(NIL);
                global.keyAttr := NIL; global.sectAttr := NIL
        END Browse;

        PROCEDURE ShowInterface* (opts: ARRAY OF CHAR);
                VAR noerr: BOOLEAN; mod, ident: DevCPT.Name; v: Views.View; title: Views.Title; t:
TextModels.Model;
                        str: ARRAY 256 OF CHAR;
        BEGIN
                GetQualIdent(mod, ident, t);
                CheckModName(mod, t);
                IF mod # "" THEN
                        OpenCompiler(mod, noerr);
                        IF noerr & (DevCPT.GlbMod[1] # NIL) & (DevCPT.GlbMod[1].name^ = mod) THEN
                                Browse(ident, opts, v, title);
                                IF v # NIL THEN
                                        IF global.clientinterface THEN Append(title, " (client interface)")
                                        ELSIF global.extensioninterface THEN Append(title, " (extension
interface)")
                                        END;
                                        Views.OpenAux(v, title)
                                END
                        ELSE str := mod$; Dialog.ShowParamMsg(modNotFoundKey, str, "", "")
                        END;
                        CloseCompiler
                ELSE Dialog.ShowMsg(noModuleNameSelectedKey)
                END;
                Kernel.Cleanup
        END ShowInterface;

        PROCEDURE ImportSymFile* (f: Files.File; OUT s: Stores.Store);
                VAR v: Views.View; title: Views.Title; noerr: BOOLEAN;
        BEGIN
                ASSERT(f # NIL, 20);
                DevCPM.file := f;
                OpenCompiler("@file", noerr);
                IF noerr THEN
                        Browse("", "", v, title);
                        s := v
                END;
                CloseCompiler;
                DevCPM.file := NIL;
                Kernel.Cleanup
        END ImportSymFile;


        (* codefile browser *)

        PROCEDURE RWord (VAR x: INTEGER);
                VAR b: BYTE; y: INTEGER;
        BEGIN
                inp.ReadByte(b); y := b MOD 256;
                inp.ReadByte(b); y := y + 100H * (b MOD 256);
                inp.ReadByte(b); y := y + 10000H * (b MOD 256);
                inp.ReadByte(b); x := y + 1000000H * b
        END RWord;

        PROCEDURE RNum (VAR x: INTEGER);
                VAR b: BYTE; s, y: INTEGER;
        BEGIN
                s := 0; y := 0; inp.ReadByte(b);
```

```
		WHILE b < 0 DO INC(y, ASH(b + 128, s)); INC(s, 7); inp.ReadByte(b) END;
		x := ASH((b + 64) MOD 128 - 64, s) + y
	END RNum;

	PROCEDURE RName (VAR name: ARRAY OF SHORTCHAR);
		VAR b: BYTE; i, n: INTEGER;
	BEGIN
		i := 0; n := LEN(name) - 1; inp.ReadByte(b);
		WHILE (i < n) & (b # 0) DO name[i] := SHORT(CHR(b)); INC(i); inp.ReadByte(b) END;
		WHILE b # 0 DO inp.ReadByte(b) END;
		name[i] := 0X
	END RName;

	PROCEDURE RLink;
		VAR x: INTEGER;
	BEGIN
		RNum(x);
		WHILE x # 0 DO RNum(x); RNum(x) END
	END RLink;

	PROCEDURE RShort (p: INTEGER; OUT x: INTEGER);
		VAR b0, b1: BYTE;
	BEGIN
		inp.ReadByte(b0); inp.ReadByte(b1);
		IF p = 10 THEN x := b0 MOD 256 + b1 * 256		(* little endian *)
		ELSE x := b1 MOD 256 + b0 * 256		(* big endian *)
		END
	END RShort;

	PROCEDURE ReadHeader (file: Files.File; VAR view: Views.View; VAR title: Views.Title);
		VAR n, i, p, fp, hs, ms, ds, cs, vs, m: INTEGER; name: Kernel.Name; first: BOOLEAN;
			text: TextModels.Model; v: TextViews.View; fold: StdFolds.Fold; d: Dates.Date; t:
Dates.Time;
			str: ARRAY 64 OF CHAR;
	BEGIN
		text := TextModels.dir.New(); global.out.ConnectTo(text);
		inp := file.NewReader(NIL);
		inp.SetPos(0); RWord(n);
		IF n = 6F4F4346H THEN
			RWord(p); RWord(hs); RWord(ms); RWord(ds); RWord(cs); RWord(vs);
			RNum(n); RName(name); title := name$; i := 0;
			WHILE i < n DO RName(imp[i]); INC(i) END;
			String("MODULE "); String(name); Ln;
			String("processor: ");
			IF p = 10 THEN String("80x86")
			ELSIF p = 20 THEN String("68000")
			ELSIF p = 30 THEN String("PPC")
			ELSIF p = 40 THEN String("SH3")
			ELSE Int(p)
			END;
			Ln;
			String("meta size: "); Int(ms); Ln;
			String("desc size: "); Int(ds); Ln;
			String("code size: "); Int(cs); Ln;
			String("data size: "); Int(vs); Ln;
			inp.SetPos(hs + ms + 12);
			RShort(p, d.year); RShort(p, d.month); RShort(p, d.day);
			RShort(p, t.hour); RShort(p, t.minute); RShort(p, t.second);
			String("compiled: ");
			Dates.DateToString(d, Dates.short, str); global.out.WriteString(str); String("  ");
			Dates.TimeToString(t, str); global.out.WriteString(str); Ln;
			Int(n); String(" imports:"); Ln;
			inp.SetPos(hs + ms + ds + cs);
			RLink; RLink; RLink; RLink; RLink; RLink; i := 0;
			WHILE i < n DO
				global.out.WriteTab; String(imp[i]); global.out.WriteTab;
				RNum(p);
				IF p # 0 THEN
					fold := StdFolds.dir.New(StdFolds.expanded, "",
TextModels.dir.New());
					global.out.WriteView(fold); Char(" "); first := TRUE;
					WHILE p # 0 DO
						RName(name); RNum(fp);
						IF p = 2 THEN RNum(m) END;
```

```
                                        IF p # 1 THEN RLink END;
                                        IF name # "" THEN
                                                IF ~first THEN String(", ") END;
                                                first := FALSE; String(name)
                                        END;
                                        RNum(p)
                                END;
                                Char(" ");
                                fold := StdFolds.dir.New(StdFolds.expanded, "", NIL);
                                global.out.WriteView(fold);
                                fold.Flip(); global.out.SetPos(text.Length())
                        END;
                        INC(i); Ln
                END
        ELSE String("wrong file tag: "); Hex(n, 8)
        END;
        v := TextViews.dir.New(text);
        v.SetDefaults(NewRuler(), TextViews.dir.defAttr);
        view := Documents.dir.New(v, width, height);
        global.out.ConnectTo(NIL);
        inp := NIL
END ReadHeader;

PROCEDURE ShowCodeFile*;
        VAR t: TextModels.Model; f: Files.File; v: Views.View; title: Views.Title; mod, ident:
DevCPT.Name;
                name: Files.Name; loc: Files.Locator; str: ARRAY 256 OF CHAR;
BEGIN
        GetQualIdent(mod, ident, t);
        IF mod # "" THEN
                str := mod$;
                StdDialog.GetSubLoc(str, "Code", loc, name);
                f := Files.dir.Old(loc, name, Files.shared);
                IF f # NIL THEN
                        ReadHeader(f, v, title);
                        IF v # NIL THEN Views.OpenAux(v, title) END
                ELSE Dialog.ShowParamMsg(modNotFoundKey, str, "", "")
                END
        ELSE Dialog.ShowMsg(noModuleNameSelectedKey)
        END
END ShowCodeFile;

PROCEDURE ImportCodeFile* (f: Files.File; OUT s: Stores.Store);
        VAR v: Views.View; title: Views.Title;
BEGIN
        ASSERT(f # NIL, 20);
        ReadHeader(f, v, title);
        s := v
END ImportCodeFile;

PROCEDURE SaveOptions*;
BEGIN
        HostRegistry.WriteBool(regKey + 'hints', options.hints);
        HostRegistry.WriteBool(regKey + 'flatten', options.flatten);
        HostRegistry.WriteBool(regKey + 'formatted', options.formatted);
        options.sflatten := options.flatten;
        options.shints := options.hints;
        options.sformatted := options.formatted
END SaveOptions;

PROCEDURE SaveGuard*(VAR par: Dialog.Par);
BEGIN
        par.disabled := (options.sflatten = options.flatten)
        & (options.shints = options.hints)
        & (options.sformatted = options.formatted)
END SaveGuard;

PROCEDURE Init*;
        VAR res: INTEGER;
BEGIN
        HostRegistry.ReadBool(regKey + 'hints', options.hints, res);
        IF res # 0 THEN options.hints := FALSE END;
        HostRegistry.ReadBool(regKey + 'flatten', options.flatten, res);
        IF res # 0 THEN options.flatten := TRUE END;
```

```
		HostRegistry.ReadBool(regKey + 'formatted', options.formatted, res);
		IF res # 0 THEN options.formatted := FALSE END;
		Dialog.Update(options);
		options.sflatten := options.flatten;
		options.shints := options.hints;
		options.sformatted := options.formatted
	END Init;

BEGIN
	root:=NIL;
	rootCount:=0;
	mode:=Undefined;
	stackCount:=0;
	stackMax:=0;

	Init;

END Robot.
```

Найти содержимое записи: +имя и +тип
+определить comp массив или запись
определить +тип статического массива (* и размерность -ПОТОМ, пока с одномерными *)

+определить pointer массив или запись
+определить тип динамического массива (* и размерность -ПОТОМ, пока с одномерными *)

+определить тип записи

+определить, является ли элемент массива записью или массивом или указателем

+рекурсивно переходить к элементам записи

+ сделать список корней, а не один корень, чтобы были модули экспрота данных как DevCPT
модульные данные
+маркер режима (запись, чтение)
+указатель на последний корневой объект (вместо списка)
+стек для рекурсий (массив указателей)
+маркер стека

+режимы работы ScanRecord

запись
+1. определение динам. массивов
+2. определение счетчиков динам массивов и занесение их размеров
+сообщения об ошибке
+нет счетчика до определения массива
+счетчик не целый
3. запись объектов
+в том же порядке, в котором заданы
+массивы записываются в цикле
+запись рекордов рекурсивно
+определить тип массива и если это запись - дать рекурсию, иначе писать атомарно
+запись рекордов масивов рекурсивно
+занести из стека в код соответствующее имя

+переменные цикла в соответствии с позицией стека
чтение

+1. определение динам. массивов
+2. определение счетчиков динам массивов
+3. чтение объектов
+чтение рекордов рекурсивно
+определить тип массива и если это запись - дать рекурсию, иначе читать атомарно
+чтение рекордов масивов рекурсивно
+занести из стека в код соответствующее имя

в тоталфеме
+сначала переделать структуру данных под минимальную
+потом добиться ее отображения в существующей программе
+потом сделать из нее генерацию кода

+в уник списке хранить указатели на корневой объект, а не строки

НАЧАЛЬНЫЕ УПРОЩЕНИЯ

+сначала задавать длины, если есть recCount:INTEGER

+потом записывать поля строго в том порядке, в котором ини приводятся

+счетчики для массивов i00, i01, i02 и т.д.

+не будет простых массиво, все динамические

+все генерации (на диск, с диска) записываем в одной процедуре

+из простых типов записываем только
INTEGER, CHAR, REAL

----------Исходный модуль----------

```
(*
        структуры данных для того, чтобы робот мог для них написать программы
        по сохранению на бинарный файл и чтению с бинарного файла

*)
MODULE  TotalfemData;
TYPE
(*
        tDictionaryCaptions=RECORD
                num*,type*:INTEGER;

                nameCount*:INTEGER;
                name*:POINTER TO ARRAY OF CHAR;

                parentNum*:INTEGER;

        END;

        tDictionaryNode*=EXTENSIBLE RECORD
                dat*:tDictionaryCaptions;
        END;


        tDictionaryData*=POINTER TO EXTENSIBLE RECORD
                nodeCount*:INTEGER;
                node*:POINTER TO ARRAY OF tDictionaryNode;
        END;
*)
(*----Recent project list-----------------------------------------------*)
        tProject*=RECORD
                nameCount*:INTEGER;
                name*:POINTER TO ARRAY OF CHAR;
        END;


        tRecentProjectData*=POINTER TO EXTENSIBLE RECORD
                projectCount*:INTEGER;
                project*:POINTER TO ARRAY OF tProject; (*sort by name*)
        END;
(*----Actions table-----------------------------------------------*)

        tActId*=RECORD
                type*,local*:INTEGER
        END;

        tActionCap*=RECORD (*make list countaining such objects to copy to array*)
                id*:tActId;

                nameCount*:INTEGER;
                name*:POINTER TO ARRAY OF CHAR

        END;

        tActionCaptionData*=POINTER TO EXTENSIBLE RECORD
```

```
                capCount*:INTEGER;
                cap*:POINTER TO ARRAY OF tActionCap
        END;


(*----Actions tree-----------------------------------------------*)

        tGoto*=RECORD
                pageNo*:INTEGER;
                id*:tActId
        END;

        tActionPageNode*=RECORD  (*make list countaining such objects to copy to array*)
                id*,parent*:tActId;
                goto*:tGoto;
        END;

        tActionPage*=RECORD  (*make list countaining such objects to copy to array*)
                pageNo*:INTEGER;

                nameCount*:INTEGER;
                name*:POINTER TO ARRAY OF CHAR;

                nodeCount*:INTEGER;
                node*:POINTER TO ARRAY OF tActionPageNode;
        END;

        tActionPagesData*=POINTER TO EXTENSIBLE RECORD
                pageCount*:INTEGER;
                page*:POINTER TO ARRAY OF tActionPage;
        END;

(*-----Data tree-------------------------------------------------*)
        tModelCaptions*=RECORD
                num*,type*:INTEGER;

                nameCount*:INTEGER;
                name*:POINTER TO ARRAY OF CHAR;
                nickCount*:INTEGER;
                nick*:POINTER TO ARRAY OF CHAR;
                nickNo*:INTEGER;

                pictureFileCount*:INTEGER;
                pictureFile*:POINTER TO ARRAY OF CHAR;

                parentNum*:INTEGER;

                xlFormFileNo*:INTEGER; (*myproject000.xls  myproject001.xls  - 000 or 001*)

                bandCount*:INTEGER;
                band*:POINTER TO ARRAY OF INTEGER (* array of band numbers like 0 1 1 1 1 1 1 2 *)
        END;

        tModelData*=POINTER TO EXTENSIBLE RECORD
                nodeCount*:INTEGER;
                node*:POINTER TO ARRAY OF tModelCaptions
        END;

(*-------------Excel template data*)
        tRamka*=RECORD
                color*:INTEGER;
                width*:REAL
        END;

        tCell*=RECORD
                type*:INTEGER; (*в зависимости от цвета. если цвет неизвестен, то белый*)
                (*толщина рамки. 1 * нет рамки. если цвет рамки 0, то читать толщину, иначе 1*)
                xlEdgeLeft*,xlEdgeRight*,xlEdgeTop*,xlEdgeBottom*:tRamka;

                fieldCount*, formatCount*,defaultCount*:INTEGER;

                field* (* ИмяМодуля.имяПеременной *),
                format* (*только для вещественных, как формат ячейки*)  ,
                default*:POINTER TO ARRAY OF CHAR
```

```
	END;

	tLine*=RECORD
		cellCount*:INTEGER;
		cell*:POINTER TO ARRAY OF tCell;
		height*:REAL
	END;

	tBand*=RECORD
		lineCount*:INTEGER;
		line*:POINTER TO ARRAY OF tLine;
		start*:INTEGER
	END;

	tTemplate*=RECORD
		bandCount*:INTEGER;
		band*:POINTER TO ARRAY OF tBand;

		widthCount*:INTEGER;
		width*:POINTER TO ARRAY OF REAL;

		nameCount*:INTEGER;
		name*:POINTER TO ARRAY OF CHAR

	END;

	tFon*=RECORD
			unchanged*,edit*,calculated*,badInput*,badOutput*,fieldMismatch*:INTEGER
	END;

	tMaketData*=POINTER TO EXTENSIBLE RECORD
		templateCount*:INTEGER;
		template*:POINTER TO ARRAY OF tTemplate;

		Fon*:tFon
	END;


END TotalfemData.
----------------------------------------Сгенерированный модуль ----------------------------------------

MODULE TotalfemDataRobot;

IMPORT Files,Stores,TotalfemData;
(*

	TYPE

 жазуу tActionCaptionData -маалымат сактагычка талапкер

 теги Data

 шартка туура келди

		tActionCaptionData = POINTER TO EXTENSIBLE RECORD
			capCount: INTEGER;
			cap: POINTER TO ARRAY OF tActionCap
		END;

 жазуу tActionPagesData -маалымат сактагычка талапкер

 теги Data

 шартка туура келди


		tActionPagesData = POINTER TO EXTENSIBLE RECORD
			pageCount: INTEGER;
			page: POINTER TO ARRAY OF tActionPage
		END;

 жазуу tMaketData -маалымат сактагычка талапкер

 теги Data
```

шартка туура келди

```
tMaketData = POINTER TO EXTENSIBLE RECORD
        templateCount: INTEGER;
        template: POINTER TO ARRAY OF tTemplate;
        Fon: tFon
END;
```

жазуу tModelData -маалымат сактагычка талапкер

теги Data

шартка туура келди

```
tModelData = POINTER TO EXTENSIBLE RECORD
        nodeCount: INTEGER;
        node: POINTER TO ARRAY OF tModelCaptions
END;
```

жазуу tRecentProjectData -маалымат сактагычка талапкер

теги Data

шартка туура келди

```
tRecentProjectData = POINTER TO EXTENSIBLE RECORD
        projectCount: INTEGER;
        project: POINTER TO ARRAY OF tProject
END;

tActId = RECORD
        type, local: INTEGER
END;

tActionCap = RECORD
        id: tActId;
        nameCount: INTEGER;
        name: POINTER TO ARRAY OF CHAR
END;

tActionPage = RECORD
        pageNo, nameCount: INTEGER;
        name: POINTER TO ARRAY OF CHAR;
        nodeCount: INTEGER;
        node: POINTER TO ARRAY OF tActionPageNode
END;

tActionPageNode = RECORD
        id, parent: tActId;
        goto: tGoto
END;

tBand = RECORD
        lineCount: INTEGER;
        line: POINTER TO ARRAY OF tLine;
        start: INTEGER
END;

tCell = RECORD
        type: INTEGER;
        xlEdgeLeft, xlEdgeRight, xlEdgeTop, xlEdgeBottom: tRamka;
        fieldCount, formatCount, defaultCount: INTEGER;
        field, format, default: POINTER TO ARRAY OF CHAR
END;

tFon = RECORD
        unchanged, edit, calculated, badInput, badOutput, fieldMismatch: INTEGER
END;

tGoto = RECORD
```

```
                pageNo: INTEGER;
                id: tActId
        END;

        tLine = RECORD
                cellCount: INTEGER;
                cell: POINTER TO ARRAY OF tCell;
                height: REAL
        END;

        tModelCaptions = RECORD
                num, type, nameCount: INTEGER;
                name: POINTER TO ARRAY OF CHAR;
                nickCount: INTEGER;
                nick: POINTER TO ARRAY OF CHAR;
                nickNo, pictureFileCount: INTEGER;
                pictureFile: POINTER TO ARRAY OF CHAR;
                parentNum, xlFormFileNo, bandCount: INTEGER;
                band: POINTER TO ARRAY OF INTEGER
        END;

        tProject = RECORD
                nameCount: INTEGER;
                name: POINTER TO ARRAY OF CHAR
        END;

        tRamka = RECORD
                color: INTEGER;
                width: REAL
        END;

        tTemplate = RECORD
                bandCount: INTEGER;
                band: POINTER TO ARRAY OF tBand;
                widthCount: INTEGER;
                width: POINTER TO ARRAY OF REAL;
                nameCount: INTEGER;
                name: POINTER TO ARRAY OF CHAR
        END;

*)
TYPE
        tActionCaptionDataRobot* = POINTER TO RECORD ( TotalfemData.tActionCaptionData );
        END;

        tActionPagesDataRobot* = POINTER TO RECORD ( TotalfemData.tActionPagesData );
        END;

        tMaketDataRobot* = POINTER TO RECORD ( TotalfemData.tMaketData );
        END;

        tModelDataRobot* = POINTER TO RECORD ( TotalfemData.tModelData );
        END;

        tRecentProjectDataRobot* = POINTER TO RECORD ( TotalfemData.tRecentProjectData );
        END;

        PROCEDURE (m: tActionCaptionDataRobot ) Memory2Binfile*(folder,file:ARRAY OF CHAR),NEW;
        VAR i,i0,i1,i2      :INTEGER;
        wr: Stores.Writer;
        loc: Files.Locator;
        f: Files.File;
        res:INTEGER;
        BEGIN
                loc := Files.dir.This(folder$);
                f:= Files.dir.New( loc, Files.dontAsk ); ASSERT( f # NIL );
                wr.ConnectTo( f );
                (*  dyn array cap *)
                (*  dyn array counter capCount *)
                IF m.cap#NIL THEN
                        m.capCount:=LEN(m.cap)
                ELSE
                        m.capCount:=0
                END;
```

```
		wr.WriteInt(m.capCount);
		FOR i0:=0 TO m.capCount-1 DO
		(* RECORD tActionCap *)
		(*  dyn array name *)
		(*  dyn array counter nameCount *)
		IF m.cap[i0].name#NIL THEN
			m.cap[i0].nameCount:=LEN(m.cap[i0].name)
		ELSE
			m.cap[i0].nameCount:=0
		END;
		(* RECORD tActId *)
		wr.WriteInt(m.cap[i0].id.type);
		wr.WriteInt(m.cap[i0].id.local);
		wr.WriteInt(m.cap[i0].nameCount);
		FOR i1:=0 TO m.cap[i0].nameCount-1 DO
			wr.WriteChar(m.cap[i0].name[i1]);
		END;
		END;
		wr.ConnectTo( NIL );
		f.Register(file$ ,'', Files.dontAsk, res );
		f.Close;
END Memory2Binfile;

PROCEDURE (m: tActionCaptionDataRobot ) Binfile2Memory*(folder,file:ARRAY OF CHAR),NEW;
VAR i,i0,i1,i2	:INTEGER;
rd: Stores.Reader;
loc: Files.Locator;
f: Files.File;
res:INTEGER;
BEGIN
		loc := Files.dir.This(folder$);
		f:= Files.dir.Old( loc, file$, Files.shared );
		rd.ConnectTo( f );
		rd.ReadInt(m.capCount);
		IF m.capCount>0 THEN
			NEW(m.cap,m.capCount)
		END;
		FOR i0:=0 TO m.capCount-1 DO
		(* RECORD tActionCap *)
		(* RECORD tActId *)
		rd.ReadInt(m.cap[i0].id.type);
		rd.ReadInt(m.cap[i0].id.local);
		rd.ReadInt(m.cap[i0].nameCount);
		IF m.cap[i0].nameCount>0 THEN
			NEW(m.cap[i0].name,m.cap[i0].nameCount)
		END;
		FOR i1:=0 TO m.cap[i0].nameCount-1 DO
			rd.ReadChar(m.cap[i0].name[i1]);
		END;
		END;
		rd.ConnectTo( NIL );
		f.Close;
END Binfile2Memory;

PROCEDURE (m: tActionPagesDataRobot ) Memory2Binfile*(folder,file:ARRAY OF CHAR),NEW;
VAR i,i0,i1,i2,i3,i4		:INTEGER;
wr: Stores.Writer;
loc: Files.Locator;
f: Files.File;
res:INTEGER;
BEGIN
		loc := Files.dir.This(folder$);
		f:= Files.dir.New( loc, Files.dontAsk ); ASSERT( f # NIL );
		wr.ConnectTo( f );
		(*  dyn array page *)
		(*  dyn array counter pageCount *)
		IF m.page#NIL THEN
			m.pageCount:=LEN(m.page)
		ELSE
			m.pageCount:=0
		END;
		wr.WriteInt(m.pageCount);
		FOR i0:=0 TO m.pageCount-1 DO
		(* RECORD tActionPage *)
```

```
		(*  dyn array name *)
		(*  dyn array counter nameCount *)
		IF m.page[i0].name#NIL THEN
			m.page[i0].nameCount:=LEN(m.page[i0].name)
		ELSE
			m.page[i0].nameCount:=0
		END;
		(*  dyn array node *)
		(*  dyn array counter nodeCount *)
		IF m.page[i0].node#NIL THEN
			m.page[i0].nodeCount:=LEN(m.page[i0].node)
		ELSE
			m.page[i0].nodeCount:=0
		END;
		wr.WriteInt(m.page[i0].pageNo);
		wr.WriteInt(m.page[i0].nameCount);
		FOR i1:=0 TO m.page[i0].nameCount-1 DO
			wr.WriteChar(m.page[i0].name[i1]);
		END;
		wr.WriteInt(m.page[i0].nodeCount);
		FOR i1:=0 TO m.page[i0].nodeCount-1 DO
		(* RECORD tActionPageNode *)
		(* RECORD tActId *)
		wr.WriteInt(m.page[i0].node[i1].id.type);
		wr.WriteInt(m.page[i0].node[i1].id.local);
		(* RECORD tActId *)
		wr.WriteInt(m.page[i0].node[i1].parent.type);
		wr.WriteInt(m.page[i0].node[i1].parent.local);
		(* RECORD tGoto *)
		wr.WriteInt(m.page[i0].node[i1].goto.pageNo);
		(* RECORD tActId *)
		wr.WriteInt(m.page[i0].node[i1].goto.id.type);
		wr.WriteInt(m.page[i0].node[i1].goto.id.local);
		END;
		END;
		wr.ConnectTo( NIL );
		f.Register(file$ ,'', Files.dontAsk, res );
		f.Close;
END Memory2Binfile;

PROCEDURE (m: tActionPagesDataRobot ) Binfile2Memory*(folder,file:ARRAY OF CHAR),NEW;
VAR i,i0,i1,i2,i3,i4		:INTEGER;
rd: Stores.Reader;
loc: Files.Locator;
f: Files.File;
res:INTEGER;
BEGIN
		loc := Files.dir.This(folder$);
		f:= Files.dir.Old( loc, file$, Files.shared );
		rd.ConnectTo( f );
		rd.ReadInt(m.pageCount);
		IF m.pageCount>0 THEN
			NEW(m.page,m.pageCount)
		END;
		FOR i0:=0 TO m.pageCount-1 DO
		(* RECORD tActionPage *)
		rd.ReadInt(m.page[i0].pageNo);
		rd.ReadInt(m.page[i0].nameCount);
		IF m.page[i0].nameCount>0 THEN
			NEW(m.page[i0].name,m.page[i0].nameCount)
		END;
		FOR i1:=0 TO m.page[i0].nameCount-1 DO
			rd.ReadChar(m.page[i0].name[i1]);
		END;
		rd.ReadInt(m.page[i0].nodeCount);
		IF m.page[i0].nodeCount>0 THEN
			NEW(m.page[i0].node,m.page[i0].nodeCount)
		END;
		FOR i1:=0 TO m.page[i0].nodeCount-1 DO
		(* RECORD tActionPageNode *)
		(* RECORD tActId *)
		rd.ReadInt(m.page[i0].node[i1].id.type);
		rd.ReadInt(m.page[i0].node[i1].id.local);
		(* RECORD tActId *)
```

```
		rd.ReadInt(m.page[i0].node[i1].parent.type);
		rd.ReadInt(m.page[i0].node[i1].parent.local);
		(* RECORD tGoto *)
		rd.ReadInt(m.page[i0].node[i1].goto.pageNo);
		(* RECORD tActId *)
		rd.ReadInt(m.page[i0].node[i1].goto.id.type);
		rd.ReadInt(m.page[i0].node[i1].goto.id.local);
		END;
		END;
		rd.ConnectTo( NIL );
		f.Close;
END Binfile2Memory;

PROCEDURE (m: tMaketDataRobot ) Memory2Binfile*(folder,file:ARRAY OF CHAR),NEW;
VAR i,i0,i1,i2,i3,i4,i5	:INTEGER;
wr: Stores.Writer;
loc: Files.Locator;
f: Files.File;
res:INTEGER;
BEGIN
		loc := Files.dir.This(folder$);
		f:= Files.dir.New( loc, Files.dontAsk ); ASSERT( f # NIL );
		wr.ConnectTo( f );
		(*  dyn array template *)
		(*  dyn array counter templateCount *)
		IF m.template#NIL THEN
			m.templateCount:=LEN(m.template)
		ELSE
			m.templateCount:=0
		END;
		wr.WriteInt(m.templateCount);
		FOR i0:=0 TO m.templateCount-1 DO
		(* RECORD tTemplate *)
		(*  dyn array band *)
		(*  dyn array counter bandCount *)
		IF m.template[i0].band#NIL THEN
			m.template[i0].bandCount:=LEN(m.template[i0].band)
		ELSE
			m.template[i0].bandCount:=0
		END;
		(*  dyn array width *)
		(*  dyn array counter widthCount *)
		IF m.template[i0].width#NIL THEN
			m.template[i0].widthCount:=LEN(m.template[i0].width)
		ELSE
			m.template[i0].widthCount:=0
		END;
		(*  dyn array name *)
		(*  dyn array counter nameCount *)
		IF m.template[i0].name#NIL THEN
			m.template[i0].nameCount:=LEN(m.template[i0].name)
		ELSE
			m.template[i0].nameCount:=0
		END;
		wr.WriteInt(m.template[i0].bandCount);
		FOR i1:=0 TO m.template[i0].bandCount-1 DO
		(* RECORD tBand *)
		(*  dyn array line *)
		(*  dyn array counter lineCount *)
		IF m.template[i0].band[i1].line#NIL THEN
			m.template[i0].band[i1].lineCount:=LEN(m.template[i0].band[i1].line)
		ELSE
			m.template[i0].band[i1].lineCount:=0
		END;
		wr.WriteInt(m.template[i0].band[i1].lineCount);
		FOR i2:=0 TO m.template[i0].band[i1].lineCount-1 DO
		(* RECORD tLine *)
		(*  dyn array cell *)
		(*  dyn array counter cellCount *)
		IF m.template[i0].band[i1].line[i2].cell#NIL THEN
			m.template[i0].band[i1].line[i2].cellCount:=LEN(m.template[i0].band[i1].line[i2].cell)
		ELSE
			m.template[i0].band[i1].line[i2].cellCount:=0
		END;
```

```
		wr.WriteInt(m.template[i0].band[i1].line[i2].cellCount);
		FOR i3:=0 TO m.template[i0].band[i1].line[i2].cellCount-1 DO
		(* RECORD tCell *)
		(*  dyn array field *)
		(*  dyn array counter fieldCount *)
		IF m.template[i0].band[i1].line[i2].cell[i3].field#NIL THEN

	m.template[i0].band[i1].line[i2].cell[i3].fieldCount:=LEN(m.template[i0].band[i1].line[i2].cell[i3].field)
		ELSE
			m.template[i0].band[i1].line[i2].cell[i3].fieldCount:=0
		END;
		(*  dyn array format *)
		(*  dyn array counter formatCount *)
		IF m.template[i0].band[i1].line[i2].cell[i3].format#NIL THEN

	m.template[i0].band[i1].line[i2].cell[i3].formatCount:=LEN(m.template[i0].band[i1].line[i2].cell[i3].forma
t)
		ELSE
			m.template[i0].band[i1].line[i2].cell[i3].formatCount:=0
		END;
		(*  dyn array default *)
		(*  dyn array counter defaultCount *)
		IF m.template[i0].band[i1].line[i2].cell[i3].default#NIL THEN

	m.template[i0].band[i1].line[i2].cell[i3].defaultCount:=LEN(m.template[i0].band[i1].line[i2].cell[i3].defau
lt)
		ELSE
			m.template[i0].band[i1].line[i2].cell[i3].defaultCount:=0
		END;
		wr.WriteInt(m.template[i0].band[i1].line[i2].cell[i3].type);
		(* RECORD tRamka *)
		wr.WriteInt(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeLeft.color);
		wr.WriteReal(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeLeft.width);
		(* RECORD tRamka *)
		wr.WriteInt(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeRight.color);
		wr.WriteReal(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeRight.width);
		(* RECORD tRamka *)
		wr.WriteInt(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeTop.color);
		wr.WriteReal(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeTop.width);
		(* RECORD tRamka *)
		wr.WriteInt(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeBottom.color);
		wr.WriteReal(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeBottom.width);
		wr.WriteInt(m.template[i0].band[i1].line[i2].cell[i3].fieldCount);
		wr.WriteInt(m.template[i0].band[i1].line[i2].cell[i3].formatCount);
		wr.WriteInt(m.template[i0].band[i1].line[i2].cell[i3].defaultCount);
		FOR i4:=0 TO m.template[i0].band[i1].line[i2].cell[i3].fieldCount-1 DO
			wr.WriteChar(m.template[i0].band[i1].line[i2].cell[i3].field[i4]);
		END;
		FOR i4:=0 TO m.template[i0].band[i1].line[i2].cell[i3].formatCount-1 DO
			wr.WriteChar(m.template[i0].band[i1].line[i2].cell[i3].format[i4]);
		END;
		FOR i4:=0 TO m.template[i0].band[i1].line[i2].cell[i3].defaultCount-1 DO
			wr.WriteChar(m.template[i0].band[i1].line[i2].cell[i3].default[i4]);
		END;
		END;
		wr.WriteReal(m.template[i0].band[i1].line[i2].height);
		END;
		wr.WriteInt(m.template[i0].band[i1].start);
		END;
		wr.WriteInt(m.template[i0].widthCount);
		FOR i1:=0 TO m.template[i0].widthCount-1 DO
			wr.WriteReal(m.template[i0].width[i1]);
		END;
		wr.WriteInt(m.template[i0].nameCount);
		FOR i1:=0 TO m.template[i0].nameCount-1 DO
			wr.WriteChar(m.template[i0].name[i1]);
		END;
		END;
		(* RECORD tFon *)
		wr.WriteInt(m.Fon.unchanged);
		wr.WriteInt(m.Fon.edit);
		wr.WriteInt(m.Fon.calculated);
		wr.WriteInt(m.Fon.badInput);
		wr.WriteInt(m.Fon.badOutput);
```

```
		wr.WriteInt(m.Fon.fieldMismatch);
		wr.ConnectTo( NIL );
		f.Register(file$ ,'', Files.dontAsk, res );
		f.Close;
	END Memory2Binfile;

	PROCEDURE (m: tMaketDataRobot ) Binfile2Memory*(folder,file:ARRAY OF CHAR),NEW;
	VAR i,i0,i1,i2,i3,i4,i5	:INTEGER;
	rd: Stores.Reader;
	loc: Files.Locator;
	f: Files.File;
	res:INTEGER;
	BEGIN
		loc := Files.dir.This(folder$);
		f:= Files.dir.Old( loc, file$, Files.shared );
		rd.ConnectTo( f );
		rd.ReadInt(m.templateCount);
		IF m.templateCount>0 THEN
			NEW(m.template,m.templateCount)
		END;
		FOR i0:=0 TO m.templateCount-1 DO
		(* RECORD tTemplate *)
		rd.ReadInt(m.template[i0].bandCount);
		IF m.template[i0].bandCount>0 THEN
			NEW(m.template[i0].band,m.template[i0].bandCount)
		END;
		FOR i1:=0 TO m.template[i0].bandCount-1 DO
		(* RECORD tBand *)
		rd.ReadInt(m.template[i0].band[i1].lineCount);
		IF m.template[i0].band[i1].lineCount>0 THEN
			NEW(m.template[i0].band[i1].line,m.template[i0].band[i1].lineCount)
		END;
		FOR i2:=0 TO m.template[i0].band[i1].lineCount-1 DO
		(* RECORD tLine *)
		rd.ReadInt(m.template[i0].band[i1].line[i2].cellCount);
		IF m.template[i0].band[i1].line[i2].cellCount>0 THEN
			NEW(m.template[i0].band[i1].line[i2].cell,m.template[i0].band[i1].line[i2].cellCount)
		END;
		FOR i3:=0 TO m.template[i0].band[i1].line[i2].cellCount-1 DO
		(* RECORD tCell *)
		rd.ReadInt(m.template[i0].band[i1].line[i2].cell[i3].type);
		(* RECORD tRamka *)
		rd.ReadInt(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeLeft.color);
		rd.ReadReal(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeLeft.width);
		(* RECORD tRamka *)
		rd.ReadInt(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeRight.color);
		rd.ReadReal(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeRight.width);
		(* RECORD tRamka *)
		rd.ReadInt(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeTop.color);
		rd.ReadReal(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeTop.width);
		(* RECORD tRamka *)
		rd.ReadInt(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeBottom.color);
		rd.ReadReal(m.template[i0].band[i1].line[i2].cell[i3].xlEdgeBottom.width);
		rd.ReadInt(m.template[i0].band[i1].line[i2].cell[i3].fieldCount);
		rd.ReadInt(m.template[i0].band[i1].line[i2].cell[i3].formatCount);
		rd.ReadInt(m.template[i0].band[i1].line[i2].cell[i3].defaultCount);
		IF m.template[i0].band[i1].line[i2].cell[i3].fieldCount>0 THEN

	NEW(m.template[i0].band[i1].line[i2].cell[i3].field,m.template[i0].band[i1].line[i2].cell[i3].fieldCount)
		END;
		FOR i4:=0 TO m.template[i0].band[i1].line[i2].cell[i3].fieldCount-1 DO
			rd.ReadChar(m.template[i0].band[i1].line[i2].cell[i3].field[i4]);
		END;
		IF m.template[i0].band[i1].line[i2].cell[i3].formatCount>0 THEN

	NEW(m.template[i0].band[i1].line[i2].cell[i3].format,m.template[i0].band[i1].line[i2].cell[i3].formatCount
)
		END;
		FOR i4:=0 TO m.template[i0].band[i1].line[i2].cell[i3].formatCount-1 DO
			rd.ReadChar(m.template[i0].band[i1].line[i2].cell[i3].format[i4]);
		END;
		IF m.template[i0].band[i1].line[i2].cell[i3].defaultCount>0 THEN
```

```
		NEW(m.template[i0].band[i1].line[i2].cell[i3].default,m.template[i0].band[i1].line[i2].cell[i3].defaultCount
)
			END;
			FOR i4:=0 TO m.template[i0].band[i1].line[i2].cell[i3].defaultCount-1 DO
				rd.ReadChar(m.template[i0].band[i1].line[i2].cell[i3].default[i4]);
			END;
			END;
			rd.ReadReal(m.template[i0].band[i1].line[i2].height);
			END;
			rd.ReadInt(m.template[i0].band[i1].start);
			END;
			rd.ReadInt(m.template[i0].widthCount);
			IF m.template[i0].widthCount>0 THEN
				NEW(m.template[i0].width,m.template[i0].widthCount)
			END;
			FOR i1:=0 TO m.template[i0].widthCount-1 DO
				rd.ReadReal(m.template[i0].width[i1]);
			END;
			rd.ReadInt(m.template[i0].nameCount);
			IF m.template[i0].nameCount>0 THEN
				NEW(m.template[i0].name,m.template[i0].nameCount)
			END;
			FOR i1:=0 TO m.template[i0].nameCount-1 DO
				rd.ReadChar(m.template[i0].name[i1]);
			END;
			END;
			(* RECORD tFon *)
			rd.ReadInt(m.Fon.unchanged);
			rd.ReadInt(m.Fon.edit);
			rd.ReadInt(m.Fon.calculated);
			rd.ReadInt(m.Fon.badInput);
			rd.ReadInt(m.Fon.badOutput);
			rd.ReadInt(m.Fon.fieldMismatch);
			rd.ConnectTo( NIL );
			f.Close;
		END Binfile2Memory;

		PROCEDURE (m: tModelDataRobot ) Memory2Binfile*(folder,file:ARRAY OF CHAR),NEW;
		VAR i,i0,i1,i2,i3,i4,i5		:INTEGER;
		wr: Stores.Writer;
		loc: Files.Locator;
		f: Files.File;
		res:INTEGER;
		BEGIN
			loc := Files.dir.This(folder$);
			f:= Files.dir.New( loc, Files.dontAsk ); ASSERT( f # NIL );
			wr.ConnectTo( f );
			(*  dyn array node *)
			(*  dyn array counter nodeCount *)
			IF m.node#NIL THEN
				m.nodeCount:=LEN(m.node)
			ELSE
				m.nodeCount:=0
			END;
			wr.WriteInt(m.nodeCount);
			FOR i0:=0 TO m.nodeCount-1 DO
			(* RECORD tModelCaptions *)
			(*  dyn array name *)
			(*  dyn array counter nameCount *)
			IF m.node[i0].name#NIL THEN
				m.node[i0].nameCount:=LEN(m.node[i0].name)
			ELSE
				m.node[i0].nameCount:=0
			END;
			(*  dyn array nick *)
			(*  dyn array counter nickCount *)
			IF m.node[i0].nick#NIL THEN
				m.node[i0].nickCount:=LEN(m.node[i0].nick)
			ELSE
				m.node[i0].nickCount:=0
			END;
			(*  dyn array pictureFile *)
			(*  dyn array counter pictureFileCount *)
```

```
		IF m.node[i0].pictureFile#NIL THEN
			m.node[i0].pictureFileCount:=LEN(m.node[i0].pictureFile)
		ELSE
			m.node[i0].pictureFileCount:=0
		END;
		(*  dyn array band *)
		(*  dyn array counter bandCount *)
		IF m.node[i0].band#NIL THEN
			m.node[i0].bandCount:=LEN(m.node[i0].band)
		ELSE
			m.node[i0].bandCount:=0
		END;
		wr.WriteInt(m.node[i0].num);
		wr.WriteInt(m.node[i0].type);
		wr.WriteInt(m.node[i0].nameCount);
		FOR i1:=0 TO m.node[i0].nameCount-1 DO
			wr.WriteChar(m.node[i0].name[i1]);
		END;
		wr.WriteInt(m.node[i0].nickCount);
		FOR i1:=0 TO m.node[i0].nickCount-1 DO
			wr.WriteChar(m.node[i0].nick[i1]);
		END;
		wr.WriteInt(m.node[i0].nickNo);
		wr.WriteInt(m.node[i0].pictureFileCount);
		FOR i1:=0 TO m.node[i0].pictureFileCount-1 DO
			wr.WriteChar(m.node[i0].pictureFile[i1]);
		END;
		wr.WriteInt(m.node[i0].parentNum);
		wr.WriteInt(m.node[i0].xlFormFileNo);
		wr.WriteInt(m.node[i0].bandCount);
		FOR i1:=0 TO m.node[i0].bandCount-1 DO
			wr.WriteInt(m.node[i0].band[i1]);
		END;
		END;
		wr.ConnectTo( NIL );
		f.Register(file$ ,'', Files.dontAsk, res );
		f.Close;
END Memory2Binfile;

PROCEDURE (m: tModelDataRobot ) Binfile2Memory*(folder,file:ARRAY OF CHAR),NEW;
VAR i,i0,i1,i2,i3,i4,i5		:INTEGER;
rd: Stores.Reader;
loc: Files.Locator;
f: Files.File;
res:INTEGER;
BEGIN
		loc := Files.dir.This(folder$);
		f:= Files.dir.Old( loc, file$, Files.shared );
		rd.ConnectTo( f );
		rd.ReadInt(m.nodeCount);
		IF m.nodeCount>0 THEN
			NEW(m.node,m.nodeCount)
		END;
		FOR i0:=0 TO m.nodeCount-1 DO
		(* RECORD tModelCaptions *)
		rd.ReadInt(m.node[i0].num);
		rd.ReadInt(m.node[i0].type);
		rd.ReadInt(m.node[i0].nameCount);
		IF m.node[i0].nameCount>0 THEN
			NEW(m.node[i0].name,m.node[i0].nameCount)
		END;
		FOR i1:=0 TO m.node[i0].nameCount-1 DO
			rd.ReadChar(m.node[i0].name[i1]);
		END;
		rd.ReadInt(m.node[i0].nickCount);
		IF m.node[i0].nickCount>0 THEN
			NEW(m.node[i0].nick,m.node[i0].nickCount)
		END;
		FOR i1:=0 TO m.node[i0].nickCount-1 DO
			rd.ReadChar(m.node[i0].nick[i1]);
		END;
		rd.ReadInt(m.node[i0].nickNo);
		rd.ReadInt(m.node[i0].pictureFileCount);
		IF m.node[i0].pictureFileCount>0 THEN
```

```
                NEW(m.node[i0].pictureFile,m.node[i0].pictureFileCount)
        END;
        FOR i1:=0 TO m.node[i0].pictureFileCount-1 DO
                rd.ReadChar(m.node[i0].pictureFile[i1]);
        END;
        rd.ReadInt(m.node[i0].parentNum);
        rd.ReadInt(m.node[i0].xlFormFileNo);
        rd.ReadInt(m.node[i0].bandCount);
        IF m.node[i0].bandCount>0 THEN
                NEW(m.node[i0].band,m.node[i0].bandCount)
        END;
        FOR i1:=0 TO m.node[i0].bandCount-1 DO
                rd.ReadInt(m.node[i0].band[i1]);
        END;
        END;
        rd.ConnectTo( NIL );
        f.Close;
END Binfile2Memory;

PROCEDURE (m: tRecentProjectDataRobot ) Memory2Binfile*(folder,file:ARRAY OF CHAR),NEW;
VAR i,i0,i1,i2,i3,i4,i5       :INTEGER;
wr: Stores.Writer;
loc: Files.Locator;
f: Files.File;
res:INTEGER;
BEGIN
        loc := Files.dir.This(folder$);
        f:= Files.dir.New( loc, Files.dontAsk ); ASSERT( f # NIL );
        wr.ConnectTo( f );
        (*  dyn array project *)
        (*  dyn array counter projectCount *)
        IF m.project#NIL THEN
                m.projectCount:=LEN(m.project)
        ELSE
                m.projectCount:=0
        END;
        wr.WriteInt(m.projectCount);
        FOR i0:=0 TO m.projectCount-1 DO
        (* RECORD tProject *)
        (*  dyn array name *)
        (*  dyn array counter nameCount *)
        IF m.project[i0].name#NIL THEN
                m.project[i0].nameCount:=LEN(m.project[i0].name)
        ELSE
                m.project[i0].nameCount:=0
        END;
        wr.WriteInt(m.project[i0].nameCount);
        FOR i1:=0 TO m.project[i0].nameCount-1 DO
                wr.WriteChar(m.project[i0].name[i1]);
        END;
        END;
        wr.ConnectTo( NIL );
        f.Register(file$ ,'', Files.dontAsk, res );
        f.Close;
END Memory2Binfile;

PROCEDURE (m: tRecentProjectDataRobot ) Binfile2Memory*(folder,file:ARRAY OF CHAR),NEW;
VAR i,i0,i1,i2,i3,i4,i5       :INTEGER;
rd: Stores.Reader;
loc: Files.Locator;
f: Files.File;
res:INTEGER;
BEGIN
        loc := Files.dir.This(folder$);
        f:= Files.dir.Old( loc, file$, Files.shared );
        rd.ConnectTo( f );
        rd.ReadInt(m.projectCount);
        IF m.projectCount>0 THEN
                NEW(m.project,m.projectCount)
        END;
        FOR i0:=0 TO m.projectCount-1 DO
        (* RECORD tProject *)
        rd.ReadInt(m.project[i0].nameCount);
        IF m.project[i0].nameCount>0 THEN
```

```
                NEW(m.project[i0].name,m.project[i0].nameCount)
            END;
            FOR i1:=0 TO m.project[i0].nameCount-1 DO
                rd.ReadChar(m.project[i0].name[i1]);
            END;
            END;
            rd.ConnectTo( NIL );
            f.Close;
        END Binfile2Memory;

END TotalfemDataRobot.
```

Чиймелер менен иштөөгө компьютерге буйрук берүү .

AutoCAD dxf interface
AutoCAD COM interface
Lib бөлчөктүү чийме чыгаруу үчүн пайдалануу мисалдары

Кагаз иштерменен иштейтурган компьютердин буйруктары .

Excel COM interface
Word COM interface
Open Office text file
LaTeX

Эрежеге салынган маалымат топтому менен иш кылуу.

FireBird
ODBC
DLL аркылуу чыгуу

Компьютерге берилген буйруктардын ыңгайлүү колдонуунун көрүнүшүн даярдоо.

GUI

Терезелердин пайдалануучунун буйрук берүү жолун иштетүү (MS Windows API)

Башка сырттагы буйрукту жүргүзүү
Башка сырттагы буйрукту токтотуу

Линукус (Linux) менен иштегендин өзгөчөлүгү.

Саясий жагы
Вирустардын себеби
Файлдардын жолдору
Чоң жана кичинекей ариптердин айрымасы
Темирлер менен файл катары иштөө
Иш бюнча мумкунчулуктор
Чоң компьютерлерди жасоо (кластерлер)

Сандык эсептөө алгоритмдери.

Чектелген Бөлчөктөрдүн Усулу (Finite Elements Method, Метод Конечных Элементов)
Lib бөлчөктүү эсеп үчүн пайдалануу мисалдары
Жумушка сунуш ким китепти окуп түшүнүп жана мисалдарды Fortran тилинен которо алса (progfe, crisfield, libnal)

Аль-Жебра жолу менен компьютерге буйруктарды берүү.

Maple

Жумушту уюштурган тартиби

кадамдарды сактоо жана өзгөртүүлөрдүн тарыхын алып баруу
иштин алдында максаттардын тизмесин белгилөө
манилүүраак нерселегре көбүраак коңүл буруу
ар бир кадамды кылаардын алдында аны жакшы түшүнүү
татаалдыкты тартип менен жеңүү
бир татаал маселени көп жеңил маселелерге бөлүү (бир чоң утулуштан көру көп майда жеңиштер оң)

Турмушта компьютерди татаал эсептер үчүн пайдалануунун мисалдары

Имараттарды Чектелген Бөлчөктөрдүн Усулу менен (Finite Elements Method) эсептөө.

Сөздүк.

Окурмандардын суроолору жана жооптор.

Урматту окурман. Эгер айта турган сөздөрүңүз же суроолоруңуз болсо KaraSandyk@rambler.ru деген дарекке кат жазыдеген дарекке кат жазыңыз.
Китеп ар дайым сиздердин суроолоруңузга жана менин жумушумга карата жаңыланып турат иншаАлла.

Китепке кошумча маалымат

дисктеги файлдар боюнча түшүнүк
интернеттеги файлдар боюнча түшүнүк

Мазмуну

Акыркы сөз.

Бул китептен үйрөнгөн нерселерди элибиз жакшылык үчун пайдалансын, түз жол менен жүрсүн, адашкандардын жолуна түшпөсүн, жана Кудая Тааланын ачуусун келтиргендердин жолуна түшпөсүн.

Омиин. Аллаху Акбар.