УНИВЕРЗИТЕТ У БЕОГРАДУ

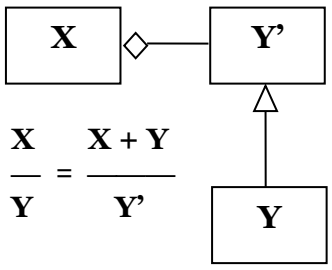
ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

Катедра за софтверско инжењерство

ОСНОВЕ ПРОГРАМИРАЊА
(скрипта- радни материјал - верзија 1.6)

Аутор:

проф. др Синиша Влајић



Београд - 2020.

Аутор
проф. др Синиша Влајић,
Наслов
ОСНОВЕ ПРОГРАМИРАЊА
(скрипта- радни материјал - верзија 1.6)
Издавач
Аутор
Рецензент
Др Владан Девеџић, ред. проф.
Припрема и дизајн
Аутор
Коректура
Аутор
Штампано
2020.

Напомена: *Електронска верзија скрипте из предмета* ***Основе програмирања*** *је бесплатна. Уколико сте у могућности донирајте Универзитетску дечију клинику, Тиршова 10, Београд. Веб адреса клинике, на којој се налазе информације везано за донацију су:*
http://tirsova.rs/lat/donacije-na-racun/

Предговор

У књизи *Основе програмирања* аутор на веома јасан и концизан начин објашњава основне концепте програмирања у програмском језику Visual Basic-у. У почетку књиге аутор уводи скоро све битне појмове који карактеришу основе рачунарства и програмирања, при чему се посебан акценат ставља на појмове хардвер и софтвер. Аутор даје неколико одличних слика где се прецизно виде елементи хардвера, софтвера и контекста програмирања. Након тога аутор објашњава типове података, променљиве и основне алгоритамске структуре (линијску, разгранату и цикличну). Приказане су процедуре и функције, где је посебна пажња усмерена на објашњење преноса параметара код процедура и функција, преко адресе и вредности. Затим аутор разматра низове матрице, слогове и датотеке, где сваку тематску јединицу прати кратка и јасна теорија са неколико репрезентативних примера. У другом делу књиге аутор објашњава основе рада са базом података помоћу Visual Basic програмског језика. У почетку је објашњен контекст база података током развоја информационих и софтверских система. Након тога је дат скуп примера којим су објашњене најзначајне наредбе SQL упитног језика за дефиницију и манипулацију подацима. На крају другог дела је размотрен концепт “скупа слогова” (recordsets) у Visual Basic-у помоћу којих се приступа и ажурира MS Access база података.
Сматрам да је књига одлично урађена и написана и са задовољством је предлажем за објављивање.

Рецензент: др Владан Девеџић, ред. проф.

Садржај

Слике

Увод

Пре пуно година, пре него што се десила "Болоња", на ФОН-у су сви студенти на првој години основних студија слушали предмет који се звао **Принципи програмирања.** Тај предмет је заједно са **Математиком I** био међу најтежим испитима за полагање. Говорило се да ко положи "Принципе" и "Математику" на првој години, тај ће готово извесно завршити Факултет. Пролазиле су године и број оних који нису положили "Принципе" је почео значајно да се повећава. Између "информатичара" и "менаџера" почело је преиспитивање да ли су "менаџерима" или да будем прецизнији, студентима који су се бавили општим менаџментом, операционим менаџментом и квалитетом, потребни Принципи програмирања у облику у коме су се тада држали. И направљен је "компромис" да се "Принципи" помере са прве године и да више не буду заједнички предмет за све студијске програме, а да се на првој години држе општи информатички предмети *Основе информационо-комуникационих технологија* и *Увод у информационе системе.* Из ове перспективе могу да кажем да су "Принципи" били сјајан предмет који је "брусио" студенте ФОН-а да буду врхунски инжењери, након завршетка основних студија. Доста наших студената, који су завршили "менаџерске" смерове, а који су слушали својевремено "Принципе" данас предаје информатику у средњим школама. Можда су "Принципи" имали неке тематске јединице које реално нису биле потребне "менаџерима", али то није, по мени, био разлог да се тај предмет потисне са прве године. Зашто то мислим, зато што свако ко жели иоле озбиљније да се бави неким "менаџерским" послом, треба да има довољно информатичких и програмерских знања, која му могу помоћи да максимално побољша и аутоматизује своје непосредно радно окружење. Није добро, да свршени студент ФОН-а, дође у ситуацију, да у фирми у којој ради мора стално некога да "моли" да му направи неки једноставан програм или упит над базом података, како би правовремено добио жељене информације. *Суштина је да ФОН-овац буде способан да самостално, брзо, квалитетно и ефикасно обради податке који су му потребни како би донео неку пословну одлуку.*

Пре пар година седео сам у ФОН-овом клубу са професором Константин Костићем, који ми је тада рекао да му је жао што "Принципи" нису више на првој години, али су он и професори са студијске групе *Операциони менаџмент* заинтересовани да се предмет сличан "Принципима" држи код њих. Тада смо се договорили да конципирамо предмет који ће да "релаксира" некадашње "Принципе" и који ће помоћи студентима да дођу до основних програмерских знања, која ће они моћи касније да користе за потребе других предмета те студијске групе. Изабрао сам да се основе програмирања уче у програмском језику Visual Basic-у [1] јер он има једноставну и "питку" синтаксу, у окружењу релационог система за управљање базом података MS Access-а. MS Access је изабран јер се развој програма над MS Access базом података ради веома брзо и једноставно. Поред тога студенти на првој година у оквиру предмета *Увод у информационе системе* уче основе MS Access-а.

У првом поглављу књиге дат је увод. У другом поглављу књиге даје се контекст програмирања са нагласком на схватање појмова рачунарство, рачунар, хардвер и софтвер. У трећем поглављу се изучавају основе програмирања. Прво су објашњени типови података и променљиве. Затим су разматране алгоритамске структуре (линијска, разграната и циклична). Након алгоритамских структура објашњене су процедуре и функције са нагласком на схватање преноса параметара преко адресе и вредности. Детаљно су разматрани низови, матрице, слогови и датотеке, уз бројне примере. У четвртом поглављу књиге описује се рад са базом података [2]. У том смислу објашњен је контекст базе података, основе SQL упитног језика и начин обраде

слогова базе података, помоћу Visual Basic концепта *скупа слогова* (*recordsets*). На крају је дата литература.

...

У ужем смислу, ова књига је намењена за студенте III године **Факултета организационих наука, Универзитета у Београду**, за предмет **Основе програмирања**, који се слуша као обавезни предмет на основним студијама, на студијском програму **Менаџмент и организација**, студијска група **Операциони менаџмент**. У ширем смислу ова књига је намењена свима онима који желе да науче основе програмирања помоћу програмског језика Visual Basic.

Аутор

1. Рачунарство

Рачунарство је научна грана која на систематски начин анализира, пројектује, имплементира и примењује рачунарске системе (рачунаре).

Рачунар је уређај који може : а) да прихвати **податке**[1] у одређеном облику б) да их обради сходно дефинисаним **наредбама** ц) и да их презентира корисницима у прихватљивом облику. Рачунар се састоји из физичког (**хардвер**) и логичког (**софтвер**) дела.

1.1 Рачунари

Савремени рачунари на којима радимо су **дигитални рачунари,** јер обрађују податке који су представљени у **дигиталном облику**. Дигитални облик указује на неки од бројних система (бинарни, декадни, октални,...) преко којих се могу представити бројеви. Иако су 40-их година прошлог века постојали покушаји у Америци да дигитални рачунари буду засновани на декадном бројном систему, рад [AWB] о логичком дизајну рачунара (који су написали Артур Буркс, Херман Голдштајн и Џон фон Нојман) дао је предност бинарном у односу на декадни бројни систем, што је била кључна одредница да будући рачунари буду засновани на **бинарном бројном систему**.

Бинарни бројни систем представља бројни систем са базом 2. То значи да се бројеви означавају преко бинарних цифара које могу имати 2 вредности 0 или 1. На пример, број 10 се бинарно представља на следећи начин: 1010.

Број 10 се добија као збир: $(2^3 * 1) + (2^2 * 0) + (2^1 * 1) + (2^0 * 0)$

Најмања организациона јединица података у рачунару је **бит**. **Бит** је бинарна цифра која може имати 0 или 1 вредност. Нпр. бинарни број 1010 се састоји из 4 бита. Најмања адресибилна јединица рачунара је **бајт** (то је најмања група битова који се могу адресирати). Бајт се састоји од 8 битова.

1.1.1 Хардвер

Хардвер *(Слика 1)* представља физички део рачунара који се састоји из компоненти. Најважније компоненте харвера су: **матична плоча**, **улазни уређаји** (тастатура, миш, скенер,...), **спољна меморија** (диск, дискета, CD,...), **оперативна (главна) меморија, централни процесор** и **излазни уређаји** (монитор, звучник, штампач,...).

Матична плоча обједињује и повезује све остале компоненте хардвера у једну целину.

Улазни уређај прихвата податке (ПД) и наредбе (НР) од корисника и шаље их до оперативне меморије.

Спољна меморија *трајно* чува податке и наредбе. Спољна меморија може да шаље и прима податке и наредбе од оперативне меморије.

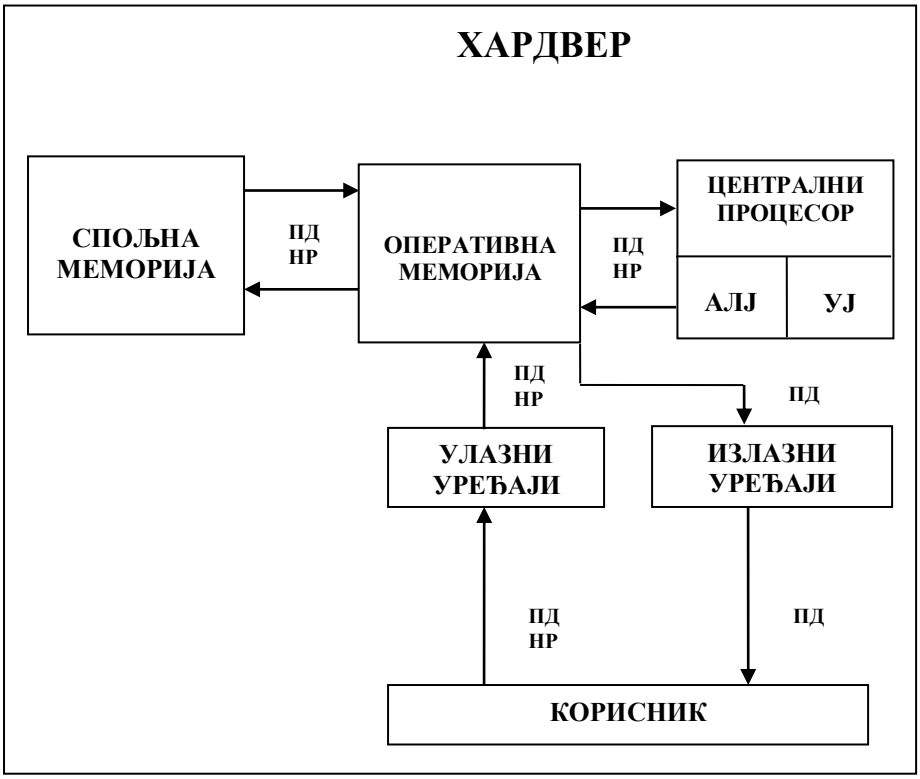
Оперативна меморија *привремено* чува податке и наредбе које је добила из спољне меморије или од крајњег корисника преко улазних уређаја. Оперативна меморија шаље податке до корисника преко излазних уређаја. Оперативна меморија шаље податке и наредбе до спољне меморије.

Централни процесор (CPU – Central Processing Unit**)** је основна компонента рачунара која управља извршењем програма и обрађује податке сходно дефинисаном програму. Централни процесор се састоји од **управљачке јединице (УЈ) и аритметичко-логичке јединице (АЛЈ). Управљачка јединица** је задужена да управља радом свих делова процесора и да синхронизује рад процесора са оперативном меморијом, улазним и излазним уређајима.

[1] **Податак** је носилац информације који је представљен у облику који је погодан за преношење, чување, или обраду, коју може да врши човек или рачунар.

Аритметичко-логичка јединица обрађује податке (изводи над подацима одговарајуће логичке и аритметичке операције) на основу наредби програма и може креирати **нове податке.** Редослед извршења наредби се контролише у **управљачкој јединици** централног процесора.
Поред АЛЈ и УЈ, централни процесор садржи **меморијске регистре** у којима се привремено чувају подаци који се обрађују и информације о тренутном стању програма који се извршава као и **кеш меморију** која садржи податке из оперативне меморије који се најчешће користе. Брзина кеш меморије је доста већа од брзине оперативне меморије.
Излазни уређај прихвата податке од оперативне меморије и шаље их до корисника.

Слика 1: Хардвер



SKP: Стандардни кораци процесора при обради наредбе

Централни процесор при обради једне наредбе програма обавља следеће стандардне кораке (основне операције):

1. Чита наредбу из оперативне меморије (**instruction fetch**).
2. Врши декодовање наредбе, како би схватио шта наредба ради и да ли постоје подаци који треба да се обраде (**instruction decode**).
3. Уколико постоје подаци чита их из оперативне меморије и учитава у регистар меморију (која се налази на централном процесору) (**operands fetch**).
4. Извршава наредбу и обрађује податке (**instruction execution**).
5. Обрађене податке уписује у оперативну меморију (**write back**).

BP: Брзина рада процесора

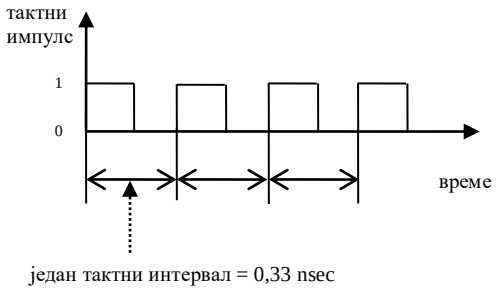
"Брзина" рада процесора се обично одређује са следеће 2 мере:
а) радним тактом (фреквенцијом) процесора.
б) бројем милиона наредби које процесор може обрадити у једној секунди.

Поред радног такта на брзину процесора утичу и **дужина процесорске речи**, **ширина магистрале података** и величина **кеш меморије**.

BP1: Радни такт процесора одређује специјално електронско коло – сат, које је обично смештено на самом процесору. Сат генерише електронске импулсе (тактне импулсе) и фреквенција (учесталост)[2] тих импулса представља радни такт процесора.

[2] Фреквенција (f) или учесталост је број осцилација у једној секунди. Период (T) је интервал времена за који се изврши једна осцилација. Фреквенција је једнака реципрочној вредности периода осциловања: f = 1 / T. Једна осцилација је еквивалент једном тактном интервалу. Период (T) је еквивалент времену једног тактног интервала.

Ако радни такт процесора P означимо са fp а време његовог тактног интервала са tp, однос између tp и fp је представљен следећом формулом:
tp = 1/fp
Уколико је радни такт процесора нпр. fp = 3GHz, то значи да сат сваке секунде генерише $3*10^9$ тактних импулса, при чему сваки тактни интервал траје tp = 1/ $3*10^9$ секунди или tp = 0,33 nsec. На сваких 0,33 nsec сат генерише један тактни импулс (Слика 2).



Радни такт процесора омогућава процесору (при вођењу извршења неког програма,) да синхронизује све делове рачунара који учествују у извршењу тог програма.
Програм се састоји из скупа наредби. За извршење једне наредбе је потребан један или више тактних интервала. Ако се нпр. програм pr1 састоји од 50 наредби, при чему је за сваку наредбу (у просеку) потребно пет тактних интервала да би се извршила, а радни такт процесора је 3GHz, тада ће време (tpr1) за извршење програма pr1 бити 82,5 nsec:

tp = 1/3GHz = 0,33nsec
tpr = 50 * 5 * 0,33 nsec = 82,5 nsec.

Уколико је радни такт процесора већи процесор ће брже извршити програм. За наведени пример уколико се повећа радни такт процесора на 4GHz, тада ће време за извршење програма pr1 бити 62,5 nsec:

tp = 1/4GHz = 0,25nsec
tpr = 50 * 5 * 0,25 nsec = 62,5 nsec.

Такође се може закључити да уколико је број тактних интервала по наредби мањи, програм ће се брже извршити.

Поред радног такта процесора, друга мера за одређивање "брзине" процесора је **број милиона наредби које процесор може обрадити у једној секунди** (MIPS - million instructions per cecond). Уколико просечан број тактних интервала по једној наредби означимо са pn, тада имамо следећу формулу:

MIPS = $(fp/pn) * 10^{-6}$

Уколико је радни такт процесора 4GHz а просечан број тактних интервала по једној наредби 5, тада се у једној секунди може извршити 800 милиона наредби:

MIPS = $(4*10^9/5) * 10^{-6} = 0,8 * 10^3 = 800$

Уколико неко повећава радни такт процесора више него што је прописао произвођач он ради **оверклоковање** (overclock), када се процесор више загрева и може да ради некоректно.

ВР2: Дужина процесорске речи

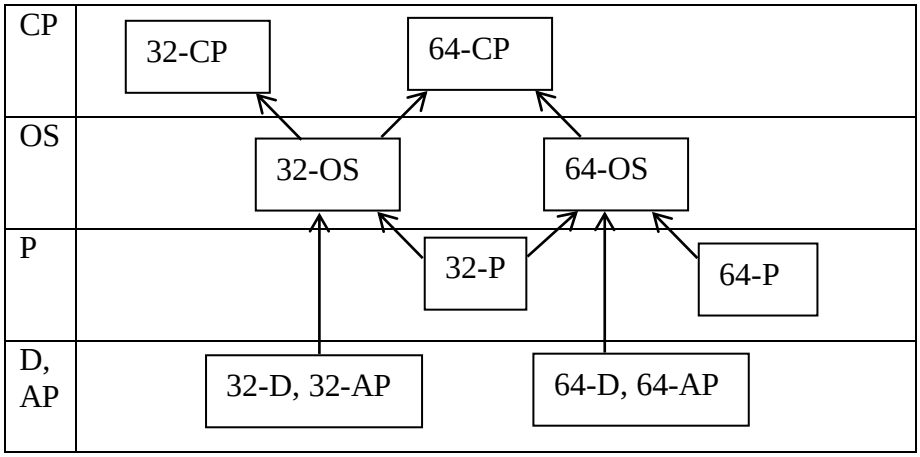
Дужина процесорске речи указује на број битова које процесор може да обради у истом тренутку. Новији процесори су углавном 32-битни и 64-битни процесори. Старији рачунари су имали 8-битне и 16-битне процесоре. Дужина процесорске речи је често ограничена величином меморијског регистра, који се налази унутар процесора. У меморијски регистар се смештају резултати међуоперација при извршењу неке наредбе.

Уколико имате 64-битни процесор пожељно је да инсталирате 64-битни оперативни систем, 64-битне програме и драјвере, како би рачунар могао да максимално искористи своје потенцијале. Рачунари који имају 64-битни процесор могу да користе 32-битне оперативне системе и програме. Старији рачунари који немају 64-битне процесоре, не могу инсталирати 64-битне оперативне системе, програме и драјвере.

32-битни оперативни систем Windows Vista Ultimate x86 може да користи максимално 3,3 GB оперативне меморије, без обзира што се на рачунару налази више оперативне меморије (нпр. 4 GB) . 64-оперативни систем може да користи максимално 192 GB оперативне меморије.

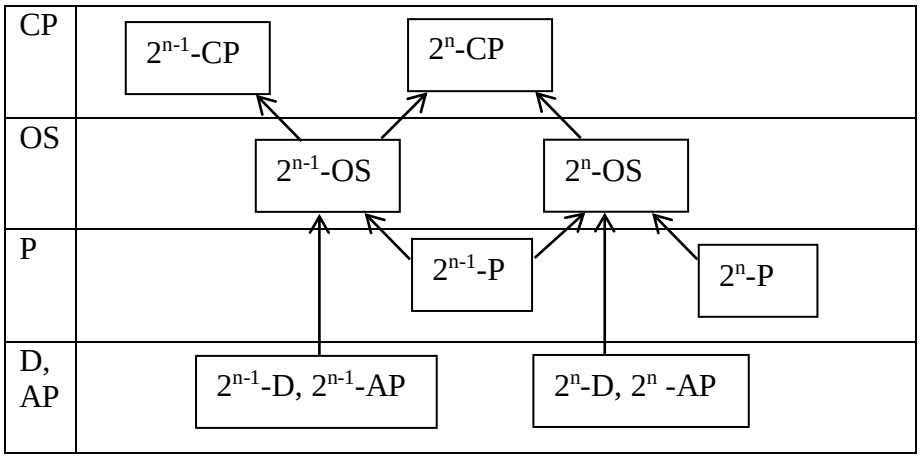
Уколико је процесор и оперативни систем 64-битни а програм 32-битни, процесор ће се понашати као 32-битни процесор.

На слици 3 се види однос 32-битних и 64-битних централних процесора (CP), оперативних система[3] (OS), програма(P), драјвера (D)[4] и антивирусних програма (AP):



[3] програм = апликативни софтвер
[4] драјвер = системски софтвер

У општем случају однос између 2^{n-1}-битних и 2^n-битних централних процесора (CP), оперативних система[5] (OS), програма(P), драјвера (D)[6] и антивирусних програма (AP), где n може бити 4,5,6,7,..., итд., може се представити на следећи начин:



ВР3: Ширина магистрале података

Магистрала представља физичку везу између компоненти хардвера, преко које се преносе:

а) подаци (магистрала података),

б) адресе где се уписују или одакле се читају подаци (адресна магистрала) и

ц) наредбе које треба да се изврше (контролна магистрала). Магистрала има свој радни такт од кога зависи брзина преноса података.

Постоје 2 врсте магистрала:

а) **системска магистрала** (FSB-front-side bus), која спаја централни процесор са оперативном меморијом и графичком картицом.

б) **улазно-излазна магистрала** која спаја централни процесор са другим хардверским компонентама.

Брзина матичне плоче је одређена радним тактом *системске магистрале*. Радни такт процесора може да буде пет пута већи од такта матичне плоче.

Ширина магистрале података се повећавала временом како су се развијали рачунари (8,16,32,64 и 128 бита). Ширина магистрале података указује на број битова који се у истом тренутку (тактном интервалу) преносе магистралом.

Уколико ширина магистрале података и дужина процесорске речи нису исте, неће се моћи користити потенцијали магистрале или процесора у потпуности. Нпр. ако је ширина магистрале података 64-бита а процесор је 32-битни, процесор неће моћи да прихвати у неком тренутку свих 64 бита који се преносе магистралом већ само 32 бита.

ВР4: Процесорска кеш меморија

Кеш меморија је веома брза меморија (у односу на оперативну меморију) у којој се налазе подаци из оперативне меморије који се најчешће користе. Она се може се налазити унутар процесора (L1) или изван њега (L2,L3). Ако при извршењу неког програма процесор успе да пронађе тражени податак, који треба да обради, у кеш меморији а не у оперативној меморији, брзина приступа податку може да буде значајно већа, него да је процесор приступио податку који је у оперативној меморији. Када процесор тражи податке он прво приступа L1 кеш меморији, која је најбржа али

[5] програм = апликативни софтвер

[6] драјвер = системски софтвер

и најмања по величини (између 16kB и 64kB). Након тога приступа L2 и L3 кеш меморији. L2 кеш меморија је величине између 256kB и 2MB и она се налази на сваком језгру процесора, уколико процесор има више језгара.L3 кеш меморија је величине између 8 и 12MB и она је заједничка меморија за сва језгра процесора.
Уколико се поређају разни типови меморије по брзини приступа подацима и њиховом капацитету (величина података) добија се приближно следеће:

типови меморије	**брзина приступа подацима**	**капацитет**
a) регистри CPU (најбржи)	1 ns	1kB
b) L1 кеш меморија	10 ns	64kB
c) L2 кеш меморија	40 ns	2MB
d) оперативна меморија	100 ns	6GB
e) хард диск	50 * 10^6 ns	1TB
f) флопи диск	95 * 10^6 ns	1.44MB
g) CD-ROM	500 * 10^6 ns	20GB
h) траке (cartige)	> 500 * 10^6 ns	10TB

1.1.2 Софтвер

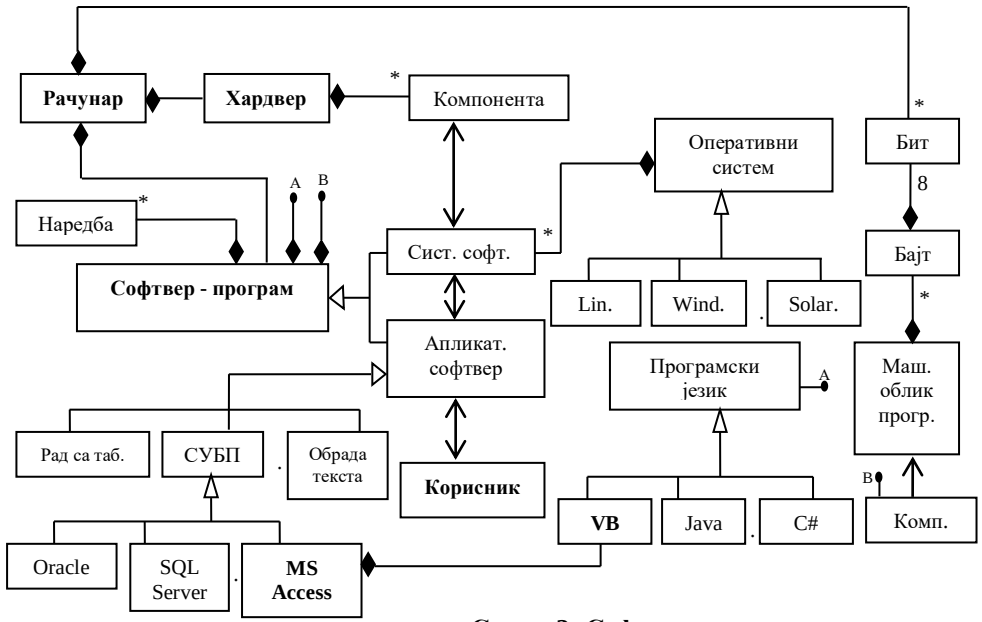
Софтвер је колекција наредби[7] која описује неки задатак или скуп задатака који треба да се изврше на рачунару како би се добио жељени резултат. Софтвер се пише у неком од програмских језика *(Слика 2)*.

Постоје две врсте софтвера:

А) **Апликативни софтвер** је програм који извршава специјализоване задатке за кориснике. Неки од апликативних софтвера су: програми за обраду текста (**Word**,Word Perfect,...), програми за рад са табелама (Excel, Lotus123,...), системи за рад са базама података, (MySQL, SQL Cerver, Oracle,MS Access,...). Апликативни софтвер нема директне везе са хардверским компонентама рачунара. Апликативни софтвер остварује везу са хардвером рачунара преко системског софтвера.

Б) **Системски софтвер** чине програми који су делови оперативног система[8] који непосредно утиче на рад хардверских компоненти рачунара. Системски софтвер омогућава везу апликативног софтвера и хардвера рачунара.

Софтвер, односно наредбе софтвера, се обично пишу у неком од **програмских језика**[9]. Да би рачунар могао да разуме наредбе, које су написане на неком од наведених виших програмских језика, потребно је да наредбе из изворног облика буду преведене у машински облик који разуме рачунар. Превођење наредби из изворног у машински (извршни) облик се ради помоћу **компајлера**[10]. Наредбе у машинском облику се састоје из скупа **битова**[11].



Слика 2: Софтвер

---

[7] **Наредбе** изводе неку радњу над подацима у току извршења програма На пример *наредба a = b*, додељује промењливој *a* вредност промењиве *b*.

[8] **Оперативни систем** представља скуп рачунарских програма који управљају хардверским и софтверским ресурсима рачунара. Главне функције оперативног система су: а) управљање процесима (апликацијама и сервисима) б) управљање меморијом ц) управљање системом датотека д) управљање мрежом е) управљање заштитом ф) управљање улазним и излазним уређајима г) управљање графичким корисничким интерфејсом (Graphical User Interfaces - GUI). Примери оперативних система су: Linux, Solaris ,...,Windows 2000 и Windows Vista.

[9] **Програмски језици** су вештачки језици, помоћу којих се праве програми који се извршавају на рачунару. Програмски језици, слично природним језицима, имају дефинисана синтаксна и семантичка правила која одређују њихову структуру и значење. Неки од програмских језика су: VB, C, C++, Java, C#, Pascal, …, Prolog.

[10] **Компајлер** је програм или скуп програма за превођење наредби изворног језика (source language) у наредбе другог - циљног језика (target language). Циљни језик може бити асемблерски језик (assembly language) или машински језик (machine language). Наредбе изворног или асемблерског програма се морају превести у наредбе машинског програма како би програм могао да се изврши на рачунару.

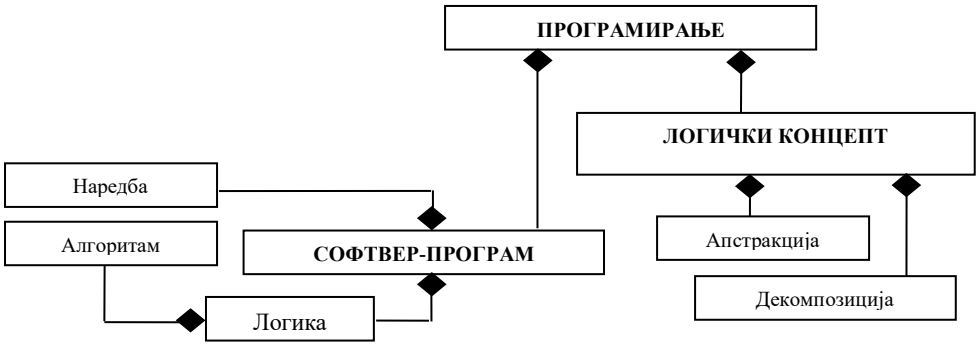
[11] **Бит** је бинарна цифра која може имати 0 или 1 вредност. Бит је најмања организациона јединица података.

Основна функција софтвера је да изврши неку обраду података. **Обрада података** обухвата следеће активности: а) памћење података у меморију рачунара преко неког од улазних уређаја рачунара, б) извођење неких операција над подацима у циљу њихове обраде и ц) читање података из меморије и њихово слање до неког од излазних уређаја рачунара. Резултат обраде података је **информација**[12]. Информација треба да има одређено значење и вредност за корисника како би он могао, сходно томе, да донесе одговарајућу **одлуку**.

[12] **Информација** је знање (скуп чињеница) које је добијено на основу испитивања или искуства у вези неке појаве. Сходно томе информације је знање које смањује или укида неодређеност у вези неке појаве. Са повећањем информација о некој појави смањује се њена ентропија. Информација се представља преко података.

2. Програмирање

Програмирање *је процес писања, тестирања и одржавања софтвера*. Логика софтвера се описује преко алгоритама[13] *(Слика 3)*. ***Алгоритам*** *је прописани скуп правила или наредби, које се извршавају у прецизно дефинисаном редоследу, за решавање неког проблема.* Два основна логичка концепта која се користе у програмирању су: **а) Апстракција** – логички процес у коме се издвајају важни аспекти неког феномена а игноришу се његови детаљи[14]. **б) Рашчлањивање (декомпозиција)** - логички процес којим се почетни проблем дели на мање делове (подпроблеме) који се независно решавају. Након тога се решења подпроблема спајају (интегришу) како би се решио почетни проблем. Декомпозиција и апстракција представљају главне логичке концепте који се користе у програмирању.

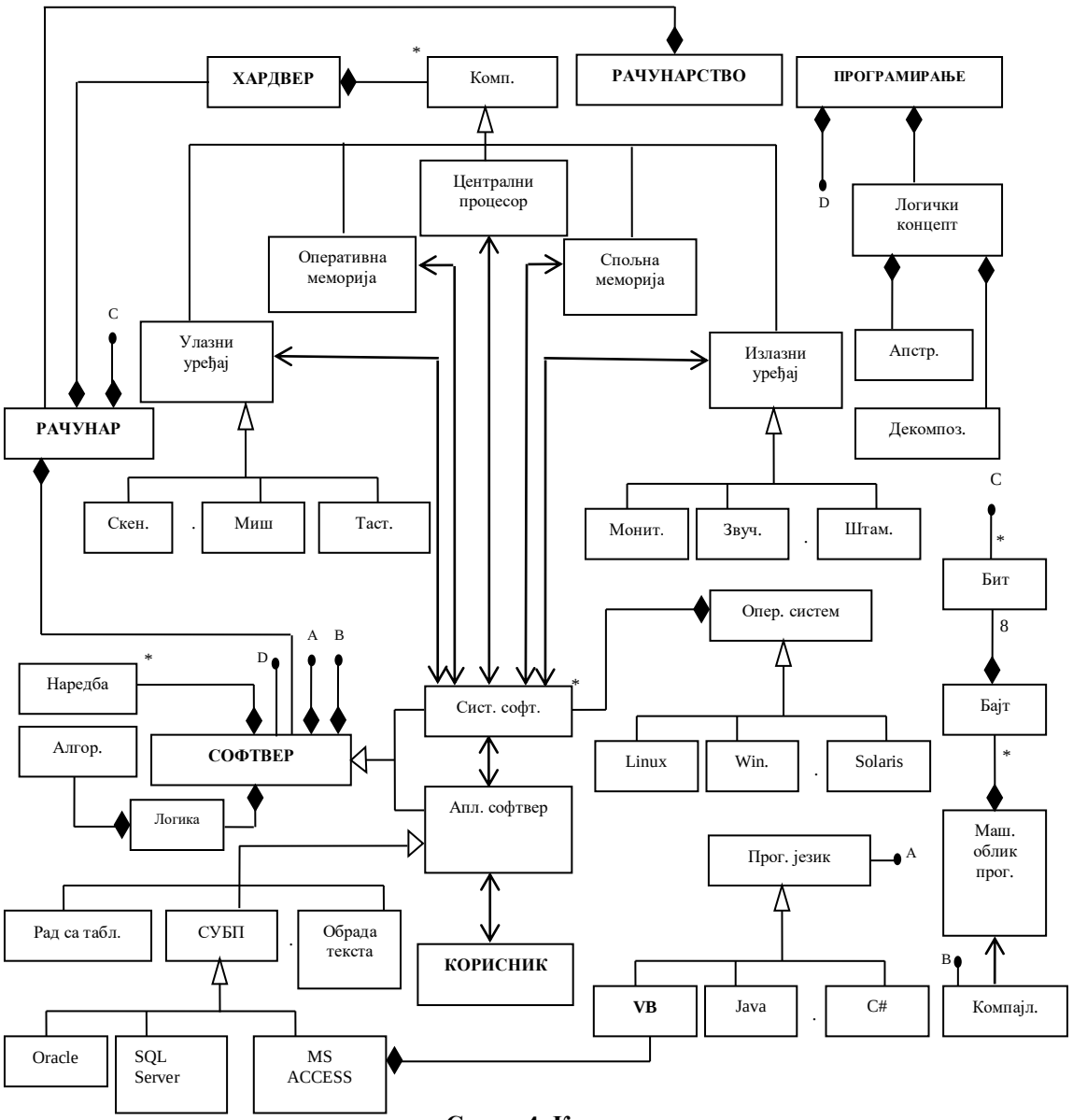


Слика 3: Програмирање

[13] Реч потиче од књиге Алгоритми индијских бројева (Algoritmi de numero Indorum), персијског математичара Алхваризимија (Al-Khwārizmī) који је живео у IX веку.

[14] У психологији апстракција је мисаони процес у коме се идеје одвајају од објеката. У филозофији апстракција је процес обликовања неког концепта.

На крају овог поглавља дајемо слику контекста програмирања *(Слика 4)*.



Слика 4: Контекст програмирања

2.1 Типови података и променљиве

Тип података представља скуп вредности и скуп операција које се могу извршити над тим вредностима. На пример, тип података *Integer* може да узме вредност из опсега од – 32,768 до 32,767. Наводимо неке од операција које се могу извршити над тим вредностима:

- Аритметичке операције (сабирање, одузимање,...)
- Логичке операције поређења (више, мање, једнако,...)

Поред *Integer* типа постоје и други типови података у VB-у као што су: *String, Boolean, Double,...*
Променљива представља појављивање неког од типова података.

Нпр:
Dim prbr As Integer;
Dim је кључна реч којом се најављује (декларише) променљива.
prbr је променљива целобројног типа.
prbr је име променљиве.

У општем смислу нека променљива се декларише на следећи начин:
Dim imePromenljive As ImeTipa

Правила везана за променљиве:

a) Име променљиве мора да започне са знаком (a-z или A-Z) или са знаком "_".
b) Име не може да садржи тачку или специјални знак.
c) Име не може да садржи бланко знак.
d) Име не може да буде дуже од 255 знакова. Препорука је да име променљиве не прелази 30 знакова.
e) Име треба да буде јединствено у својој области важења (scope).

Више променљивих може да се декларише на следеће начине:
Dim imePromenljive1 As ImeTipa1
Dim imePromenljive2 As ImeTipa2
Dim imePromenljive3 As ImeTipa3

или

Dim imePromenljive1 As ImeTipa1, Dim imePromenljive2 As ImeTipa2
Dim imePromenljive3 As ImeTipa3

2.2 Алгоритамске структуре

Алгоритам је прописани скуп правила или наредби, које се извршавају у прецизно дефинисаном редоследу, за решавање неког проблема. Постоје три елементарне алгоритамске структуре:
а) Линијска алгоритамска структура
б) Разграната алгоритамска структура
ц) Циклична алгоритамска структура

ЛАС - Линијска алгоритамска структура

При једном пролазу програма кроз линијску алгоритамску структуру, извршава се једна или више наредби, само једном, у редоследу у коме су дефинисане.

Кориснички захтев ЛАС1: *Написати програм који ће преко улазне функције InputBox() прихватити број а затим га приказати преко функције MsgBox.*

Private Sub **Command0_Click**()	Почетак процедуре.
Dim br As Integer	Декларација променљиве *br*.
' Линијска алгоритамска структура	Коментар у програму (све у истом реду иза апострофа ' је коментар).
br = InputBox("Unesi broj:")	Корисник уноси вредност преко улазне функције *InputBox()*, са којом се пуни променљива *br*.
MsgBox (br)	Приказ вредности на излазу -"боксу поруке" (*message box*).
End Sub	Крај процедуре

Питања:

1. *Који је тип променљиве br?*
2. *Који је скуп могућих вредности променљиве br?*
3. *Које се операције могу извршити над променљивом br?*
4. *Чему служи наредба InputBox?*
5. *Чему служи наредба MsgBox?*
6. *Колико се пута извршавају наредбе код линијске алгоритамске структуре при једном пролазу програма?*

РАС - Разграната алгоритамска структура

При једном пролазу програма кроз разгранату алгоритамску структуру од више група наредби које се налазе у различитим гранама, изабира се само једна од грана и извршавају се наредбе у тој грани, само једном, док се остале гране неће извршити ни једном. Избор гране зависи од задовољености услова који је постављен на почетку гране.

Кориснички захтев РАС1: *Написати програм који ће преко улазне функције InputBox() прихватити број а затим преко функције MsgBox приказати да ли је број паран или непаран.*

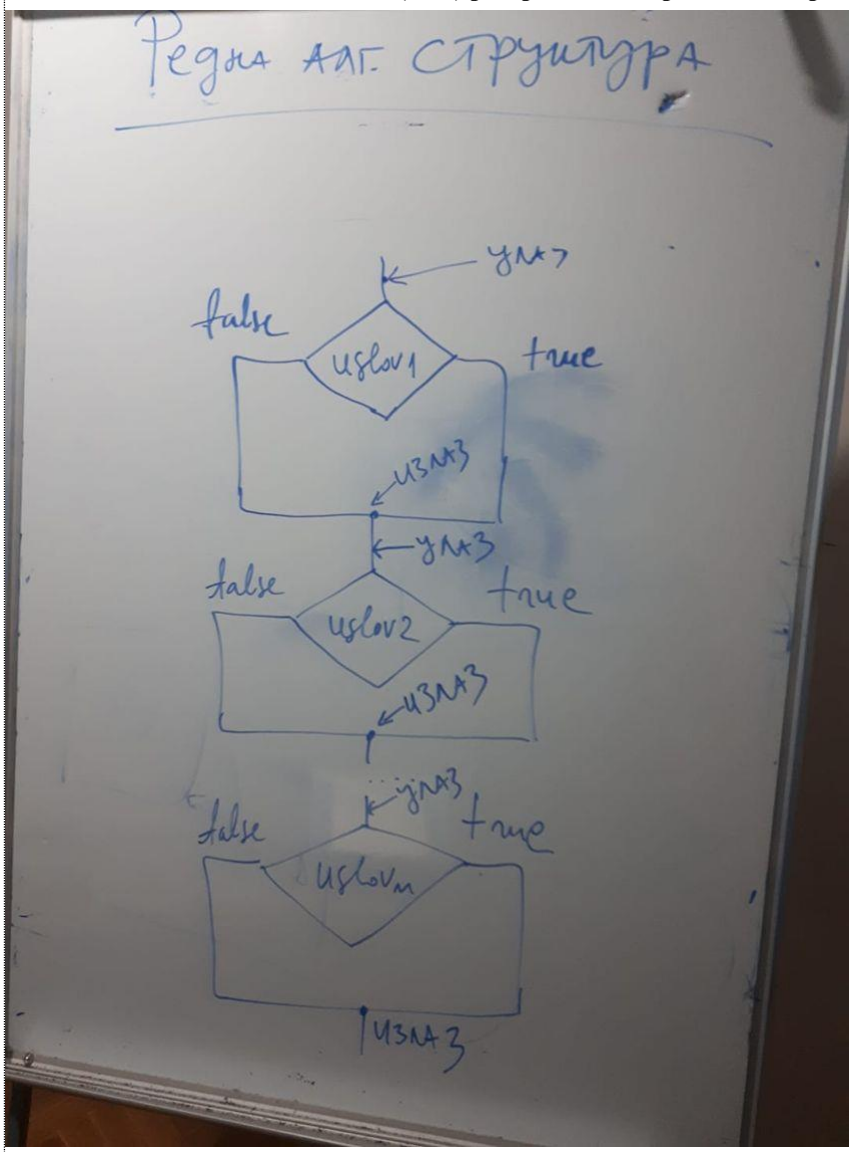
Private Sub **Command1_Click**()	Почетак процедуре.
Dim br As Integer	Декларација променљиве *br*.
br = InputBox("Unesi broj:")	Корисник уноси вредност преко улазне функције *InputBox(),* са којом се пуни променљива *br*.
‘ Разграната алгоритамска структура	Коментар у програму.
If (br Mod 2) = 0 Then	Почетак разгранате структуре. Ако је број *br* паран, тада се
MsgBox (br & " je paran broj.")	приказује порука да је број паран,
Else	иначе се,
MsgBox (br & " je neparan broj.")	приказује порука да је број непаран.
End If	Крај разгранате структуре.
End Sub	Крај процедуре.

Постоје два типа разгранатих алгоритамских структура:
а) редне разгранате алгоритамске структуре
б) паралелне разгранате алгоритамске структуре

Код **редне разгранате алгоритамске структуре** имамо две или више разгранатих алгоритамских структура које су повезане тако да излаз из једне разгранате алгоритамске структуре представља улаз у другу разгранату алгоритамску структуру. Уколико имамо n разгранатих алгоритамских структура, улаз у наведену структуру је код прве разгранате алгоритамске структуре, док је излаз, код последње разгранате алгоритамске структуре. Свака од наведених разгранатих структура има произвољан број грана.

Општи облик редне разгранате алгоритамске структуре (*Слика 5*) са n разгранатих алг. структура (***if–then–else-endif*** облика) је:

```
if uslov1  then    --------> улаз у прву разгранату алгоритамску структуру
     ...
else
     ...
end if             --------> излаз из прве разгранате алгоритамске структуре

if uslov2  then --------> улаз у другу разгранату алгоритамску структуру
     ...
else
     ...
end if             --------> излаз из друге  разгранате алгоритамске структуре
...
if uslovn  then --------> улаз у последњу (n-ту) разгр. алгоритамску структуру
     ...
else
     ...
end if --------> излаз из последње (n-те) разгранате алгоритамске стр.
```



Слика 5: Општи облик редне разгранате алгоритамске структуре

Општи облик редне разгранате алгоритамске структуре са n разгранатих алгоритамских структура (***select–case-endselect*** облика) је:

```
Select Case vrednostKojaSePoredi ----> улаз у прву разгранату алг. структуру

Case uslov1
        ...
Case uslov2
        ...
Case Else
        ...
End Select   --------> излаз из прве разгранате алг.структуре
...

Select Case vrednostKojaSePoredi --------> улаз у другу разгр. алг. структуру

Case uslov1
        ...
Case uslov2
        ...
Case Else
        ...
End Select   --------> излаз из друге  разгранате алгоритамске структуре
...
Select Case vrednostKojaSePoredi -----> улаз у последњу (n-ту) разгр. алг. структуру
Case uslov1
        ...
Case uslov2
        ...
Case Else
        ...
End Select   ----> излаз из последње (n-те) разгранате алгоритамске стр.
```

Паралелне разгранате алгоритамске структуре имају једну основну разгранату алгоритамску структуру, унутар које се налазе (уколико постоје) остале разгранате алгоритамске структуре. Код паралелених разгранатих алгоритамских структура постоји један улаз и један излаз из те структуре. Уколико се користи ***if–then–else-endif*** облик разгранате алгоритамске структуре, основна алгоритамска структура има 2 гране.

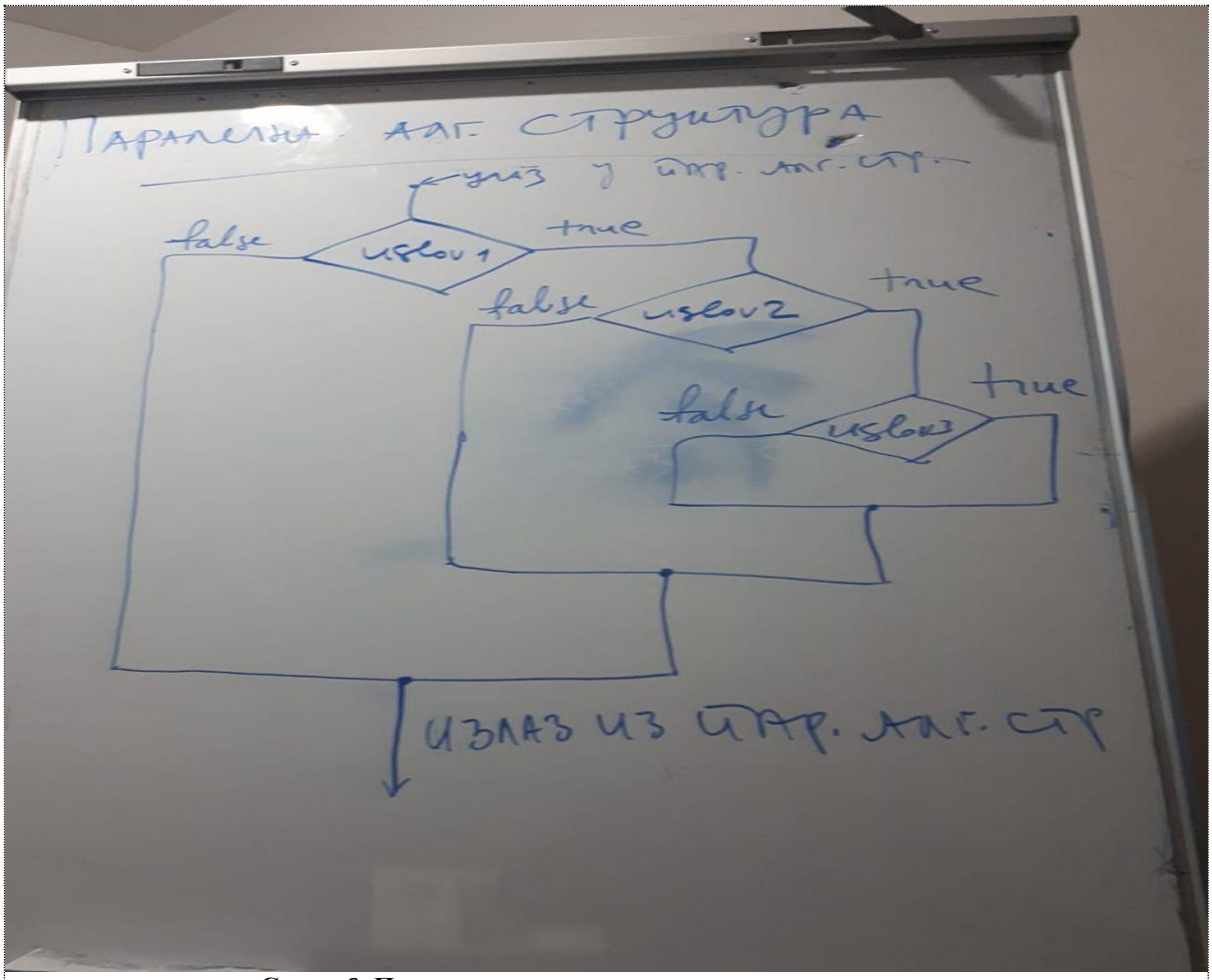
```
if uslov1  then --------> улаз у паралелну разгр. алг. структуру
...
else
...
end if  --------> излаз из паралелне разгранате  алгоритамске структуре
```

Уколико се користи ***select–case-endselect*** облик разгранате алгоритамске структуре, основна алгоритамска структура може имати више грана *(n)*.

```
Select Case vrednostKojaSePoredi -----> улаз у паралелну разгр. алг. структуру

Case uslov1
        ...
Case uslov2
        ...
Case uslovn
          ...
Case Else
        ...
End Select   --------> излаз из паралеле разг. алг. структуре
```

Уколико код паралелне разгранате алгоритамске структуре if_1, у свакој од грана основне разгранате алгоритамске структуре, имамо неке друге паралелне разгранате алгоритамске структуре if_2, if_3, ..., if_n, тада је број грана паралелне разгранате алгоритамске структуре if_1 једнак броју услова (*n*) те структуре плус један (*Слика 6*). На пример:

```
if uslov1  then      --------> if1 паралелна алгоритамска структура – почетак основне алг. стр.

        if uslov2  then  --------> if2 паралелна алгоритамска структура

                if uslov3  then --------->  if3 паралелна алгоритамска структура
                ...          --------> прва грана
                else
                ...          --------> друга грана
                end if
        else
        ...                  --------> трећа грана
        end if
else
...                          --------> четврта грана
end if       --------->  крај основне разгранате алгоритамске структуре
```



Слика 6: Паралелна разграната алгоритамска структура

Уколико имамо три услова у структури if1 имаћемо укупно 4 гране.

На пример:

Select Case *vrednostKojaSePoredi* --> **if1 паралелна алг. структура – почетак основне --> алгоритамске структуре.**
Case uslov1
    if uslov2 then --------> if2 паралелна алгоритамска структура
    ... --------> прва грана
    else
    ... --------> друга грана
    end if
Case uslov3
    ... --------> трећа грана
Case uslov4
    if uslov5 then --------> if3 паралелна алгоритамска структура
    ... --------> четврта грана
    else
    ... --------> пета грана
    end if
Case Else
    ... --------> шеста грана
End Select --------> **крај основне разгранате алгоритамске структуре**

Уколико имамо пет услова у структури if1 имаћемо укупно 6 грана.

Питања:

1. *Чему служи оператор Mod?*
2. *Који је општи облик редне разгранате алгоритамске структуре?*
3. *Колико се пута наредбе изабране гране извршавају код разгранате алгоритамске структуре при једном пролазу програма?*
4. *Објаснити разлику између редних и паралелних разгранатих алгоритамских структура.*
5. *Код паралелених разгранатих алгоритамских структура која веза постоји између броја услова те структуре и броја грана?*

ЦАС - Циклична алгоритамска структура

При једном пролазу програма кроз цикличну алгоритамску структуру, извршава се једна или више наредби, једном или више пута у зависности од задовољености услова за извршење циклуса, у редоследу у коме су дефинисане. Уколико се броје пролази (циклуси) кроз цикличну алгоритамску структуру (почев од првог пролаза), потребно је да се дефинише **индекс циклуса** који ће да прати број текућег пролаза (први пролаз, други пролаз, ...) кроз цикличну алгоритамску структуру.

Кориснички захтев ЦАС1: *Написати програм у коме ће се напунити целобројна низовна променљива преко улазне Функције InputBox(). Након тоге приказати наведену низовну променљиву преко функције MsgBox().*

Private Sub **Command2_Click**()	Почетак процедуре (програма).
Dim n(5) As Integer	Декларисање низа од 5 елемената целобројног типа.
Dim i As Integer	Декларисање целобројне променљиве.
MsgBox ("Unesite niz od 5 brojeva:")	Приказ поруке кориснику програма.
' Цикличне алгоритамске структуре	Коментар у програму.
For i = 1 To 5	Прва циклична алг. структура – почетак. Индекс циклуса i је на почетку постављен на 1.
n(i) = InputBox("Unesi " & i & ". broj:")	Корисник уноси вредности елементима низа.
Next i	Вредност индекса циклуса i се повећава за један. Када i добије 6 излази се из цикличне структуре.
For i = 1 To 5	Друга циклична алгоритамска структура – почетак. Индекс циклуса i је постављен на 1.
MsgBox (i & ". broj je: " & n(i))	Приказ вредности елемената низа на излазу .
Next i	Вредност индекса циклуса i се повећава за један. Када i добије 6 излази се из цикличне структуре.
End Sub	Крај процедуре (програма).

Програм се извршава и пролази кроз наредбе. Вредности променљивих програма, током проласка програма кроз наредбе прве цикличне алгоритамске структуру су:

Циклус (пролаз)	**i**	**n(i) = InputBox("Unesi " & i & ". broj:")**
Први	1	n(1) = InputBox("Unesi 1. broj:")
Други	2	n(2) = InputBox("Unesi 2. broj:")
Трећи	3	n(3) = InputBox("Unesi 3. broj:")
Четврти	4	n(4) = InputBox("Unesi 4. broj:")
Пети	5	n(5) = InputBox("Unesi 5. broj:")

У сваком пролаза, кроз прву цикличну структуру, сваки елемент низа (почев од првог елемента низа) добија неку вредност, преко улазне функције InputBox():

n(i)
n(1) = 7
n(2) = 2
n(3) = 3
n(4) = 5
n(5) = 9

Вредности промењљивих програма, током проласка програма кроз наредбе друге цикличне алгоритамске структуру су:

Циклус (пролаз)	i	MsgBox (i & ". broj je: " & n(i))
Први	1	MsgBox (1. broj je: 7)
Други	2	MsgBox (2. broj je: 2)
Трећи	3	MsgBox (3. broj je: 3)
Четврти	4	MsgBox (4. broj je: 5)
Пети	5	MsgBox (5. broj je: 9)

У сваком пролаза, кроз другу цикличну структуру, сваки елемент низа (почев од првог елемента низа) приказује своју вредност на излазу, преко излазне функције MsgBox():

Излаз
n(1) = 7
n(2) = 2
n(3) = 3
n(4) = 5
n(5) = 9

Питања:

1. *Да ли се код низа унапред зна број елемената низа?*
2. *Да ли су елементи низа истог или различитог типа?*
3. *Колико се пута извршавају наредбе код цикличне алгоритамске структуре при једном пролазу програма?*
4. *Дати дефиницију низовног типа који има 7 елемената који су реални бројеви.*
5. *Како се приступа елементима низа?*

Типови цикличних алгоритамских структура

Постоје 4 типа цикличне алгоритамске структуре:
а) Ради ... док није
б) Ради ... док је
ц) Док није ... ради
д) Док је ... ради

ЦАС1: Ради ... док није

Циклична алгоритамска структура (РАДИ ... ДОК НИЈЕ) се извршава док није задовољен услов да је индекс циклуса већи од неке унапред задате вредности. Ова циклична структура се користи када смо сигурни да је први пролаз кроз циклус известан и без услова.

Private Sub **Command3_Click**()	Почетак процедуре (програма).
Dim n(5) As Integer	Декларисање низа од 5 елемената целобројног типа.
Dim i As Integer	Декларисање целобројне променљиве.
MsgBox ("Unesite niz od 5 brojeva:")	Приказ поруке кориснику програма.
i = 1	Индекс циклуса *i* је на почетку постављен на 1.
Do	Циклична алг. структура (**ради-док није**) – почетак.
n(i) = InputBox("Unesi " & i & ". broj:")	Корисник уноси вредности елементима низа.
i = i + 1	Повећање индекса циклуса за 1.
Loop Until i > 5	Док није индекс циклуса већи од 5 извршава се циклична алг. структура, у супротном излази се из цикличне алг. структуре.
For i = 1 To 5	Друга циклична алг. структура – почетак. Индекс циклуса *i* је на почетку постављен на 1.
MsgBox (i & ". broj je: " & n(i))	Приказ вредности елемената низа на излазу .
Next i	Вредност индекса циклуса *i* се повећава за један. Када *i* добије 6 излази се из цикличне алг. структуре.
End Sub	Крај процедуре (програма).

ЦАС2: Ради ... док је

Циклична алгоритамска структура (РАДИ ... ДОК ЈЕ) се извршава док је задовољен услов да је индекс циклуса мањи од неке унапред задате вредности. Ова циклична структура се користи када смо сигурни да је први пролаз кроз циклус известан и без услова.

Private Sub Command4_Click()	
Dim n(5) As Integer	
Dim i As Integer	
MsgBox ("Unesite niz od 5 brojeva:")	
i = 1	Индекс циклуса *i* је на почетку постављен на 1.
Do	Циклична алгоритамска структура (**ради-док је**) – почетак.
n(i) = InputBox("Unesi " & i & ". broj:")	Корисник уноси вредности елементима низа.
i = i + 1	Повећање индекса циклуса за 1.
Loop While i < 6	Док је индекс циклуса мањи од 6 извршава се циклична алгоритамска структура, у супротном излази се из цикличне алгоритамске структуре.
For i = 1 To 5	
MsgBox (i & ". broj je: " & n(i))	
Next i	
End Sub	

ЦАС3: Док није ... ради

Циклична алгоритамска структура (ДОК НИЈЕ ... РАДИ) се извршава док није задовољен услов да је индекс циклуса већи од неке унапред задате вредности.

Private Sub Command5_Click()	
Dim n(5) As Integer	
Dim i As Integer	
MsgBox ("Unesite niz od 5 brojeva:")	
i = 1	Индекс циклуса *i* је на почетку постављен на 1.
Do Until i > 5	Циклична алгоритамска структура (**док није - ради**) – почетак. Док није задовољен услов да је индекс циклуса i већи од 5 извршава се циклична алгоритамска сруктура, у супротном излази се из цикличне алгоритамске структуре.
n(i) = InputBox("Unesi " & i & ". broj:")	Корисник уноси вредности елементима низа.
i = i + 1	Повећање индекса циклуса за 1.
Loop	Крај цикличне алгоритамске структуре.
For i = 1 To 5	
MsgBox (i & ". broj je: " & n(i))	
Next i	
End Sub	

ЦАС4: Док је ... ради

Циклична алгоритамска структура (ДОК ЈЕ ... РАДИ) се извршава док је задовољен услов да је индекс циклуса мањи од неке унапред задате вредности.

Private Sub Command6_Click()	
Dim n(5) As Integer	
Dim i As Integer	
MsgBox ("Unesite niz od 5 brojeva:")	
i = 1	Индекс циклуса *i* је на почетку постављен на 1.
Do While i < 6	Циклична алг. структура (**док је - ради**) – почетак. Док је задовољен услов да је индекс циклуса *i* мањи од 6 извршава се циклична алг. сруктура, у супротном излази се из цикличне алг. структуре.
n(i) = InputBox("Unesi " & i & ". broj:")	Корисник уноси вредности елементима низа.
i = i + 1	Повећање индекса циклуса за 1.
Loop	Крај цикличне алг. структуре.
For i = 1 To 5	
MsgBox (i & ". broj je: " & n(i))	
Next i	
End Sub	

Питања:

1. *Чему служи услов у циклусу?*
2. *Шта је индекс циклуса (дати пример) ?*
3. *Зашто се индексу циклуса пре почетка извршења циклуса додељује почетна вредност?*
4. *Напунити елементе низа и приказати након тога вредност сваког другог елемента низа.*

2.3 Процедуре

Процедура представља издвојени скуп наредби у програму која се извршава у циљу решавања неког задатка (проблема). Процедуре се обично чувају у софтверским библиотекама како би могле поново да се користе у различитим програмима. Процедуре за разлику од функција не могу да врате вредност.

Процедура се дефинише на следећи начин:

```
Sub ImeProcedure(par1 as NekiTipPodatka,par2 as NekiTipPodatka,...)
    ' telo procedure
End Sub
```

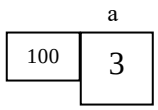
Sub представља кључну реч која указује на почетак дефинисања процедуре. Свака процедура има јединствено име (*ImeProcedure*) у модулу. Иза назива процедуре дефинише се листа параметара процедуре (*par1,par2,...*). Листа параметара може да буде празна. Сваки од параметара је везан за неки од типова података. У телу процедуре се уносе наредбе које одређују шта ће се десити у процедури. На крају се ставља *End Sub* кључне речи које указују на крај процедуре.

Пренос параметара преко адресе

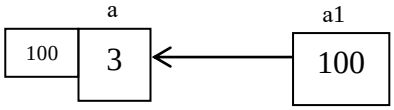
Параметри код процедура могу да се преносе преко адресе (референце) и преко вредности. Када се параметри **преносе преко адресе**, промене које се изврше на параметрима у позивајућој процедури *PP()* ће се видети у главној процедури *GP()* која позива процедуру *PP()*.

Sub **GP**()	почетак главне процедуре *GP()* (одакле почиње програм)
Dim a As Integer	Декларисање целобројне променљиве *a*.
a =3	Додељивање вредности 3 променљивој *a*.
PP a	Позив процедуре *PP()* којој се прослеђује параметар *a*. Код позива процедуре параметар a се назива **стварни параметар.**
MsgBox (a)	Приказ вредности променљиве *a*.
End Sub	Крај главне процедуре
Sub **PP**(ByRef a1 as Integer)	Почетак процедуре *PP()*, параметар *a1* се преноси преко адресе. На месту где је дефинисана процедура, параметар *a1* се назива **формални параметар.** Уколико се ништа не постави испред параметра процедуре тада се подразумева да је пренос параметра преко адресе. Уколико желимо експлицитно да нагласимо да желимо да пренос параметара буде преко адресе наводи се кључна реч ***ByRef***. нпр. *Sub PP(**ByRef** a1 as Integer)*
a1 = 5	Додељивање вредности 5 параметру a1.
End Sub	Крај процедуре *PP()*.

Када главна процедура *GP()* почне да се извршава тада променљива *a* добија вредност 3. Променљива *a* се налази на некој фиктивној адреси 100.



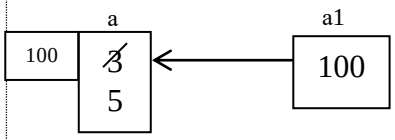
Променљива *a* се прослеђује као стварни параметар при позиву процедуре *PP()*. Будући да је у питању пренос параметара преко адресе, променљива *a* прослеђује формалном параметру *a1* у процедури *PP( )* своју адресу (100) а не вредност (3).



Све што се касније у процедури *PP()* буде радило са параметром *a1* имаће директан утицај на променљиву *a* из главне процедуре *GP()*. Тако наредба

a1 = 5

има следећи утицај на a:



Након извршења процедуре *PP()* контрола програма се враћа у главну процедуру *GP()*. Тада се извршава наредба:

MsgBox (a)

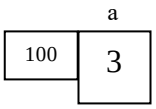
Као ефекат наведене наредбе на излазу ће бити приказана вредност 5 од променљиве *a*.

Пренос параметара преко вредности

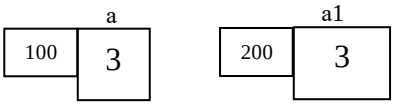
Када се параметри **преносе преко вредности**, промене које се изврше на параметрима у позивајућој процедури *PP()* неће се видети у главној процедури *GP()* која позива процедуру *PP()*.

Sub **GP**()	Почетак главне процедуре *GP()*(одакле почиње програм).
Dim a As Integer	Декларисање целобројне пронељиве *a*.
a =3	Додељивање вредности 3 променљивој *a*.
PP a	Позив процедуре *PP()* којој се прослеђује параметар *a*.
MsgBox (a)	Приказ вредности променљиве *a*.
End Sub	Крај главне процедуре
Sub **PP**(ByVal a1 as Integer)	Почетак процедуре *PP()*, параметар *a1* се преноси преко вредности. Испред формалног параметра *a1* се ставља кључна реч *ByVal* која одређује да ће се параметар преносити преко вредности.
a1 = 5	Додељивање вредности 5 параметру *a1*.
End Sub	Крај процедуре *PP()*.

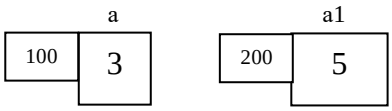
Када главна процедура *GP()* почне да се извршава тада променљива *a* добија вредност 3. Променљива *a* се налази на некој фиктивној адреси 100.



Променљива *a* се прослеђује као стварни параметар при позиву процедуре *PP()*. Будући да је у питању пренос параметара преко вредности, променљива *a* прослеђује формалном параметру *a1* у процедури *PP( )* своју вредност (3) а не адресу (100). Параметар *a1* постаје копија променљиве *a,* која се налази на посебној адреси (фиктивно адреса је 200).



Све што се касније у процедури *PP()* буде радило са параметром *a1* **неће имати директан утицај** на променљиву *a* из главне процедуре *GP()*. Тако наредба
a1 = 5, имаће утицај на параметар *a1*, али не и на параметар *a:*



То значи да процедура *PP()* неће имати утицај на променљиву *a* главне процедуре *GP()*.
Након извршења процедуре *PP()* контрола програма се враћа у главну процедуру *GP()*. Тада се извршава наредба:

```
MsgBox (a)
```

Као ефекат наведене наредбе на излазу ће бити приказана вредност 3 од променљиве *a*.

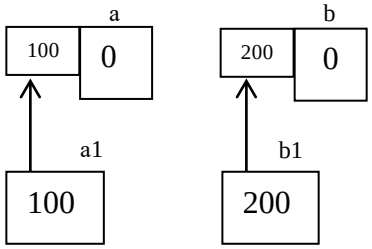
Примери:

Кориснички захтев ПР1: *Написати програм који ће преко улазне функције InputBox() прихватити број а затим преко функције MsgBox приказати тај број. Програм урадити преко процедура.*

Private **Sub Comand0_Click**()	Почетак главне процедуре.
Dim a As Integer	Декларисање целобројне променљиве *a*.
Dim b As Integer	Декларисање целобројне променљиве *b*.
unesiBrojeve a, b	Позив процедуре *unesiBrojeve()* којој се прослеђују адресе променљивих *a* и *b*.
prikaziBrojeve a, b	Позив процедуре *prikaziBrojeve()* којој се прослеђују вредности променљивих *a* и *b*.
End Sub	Крај главне процедуре.
Sub **unesiBrojeve**(a1 As Integer, b1 As Integer)	Почетак процедуре *unesiBrojeve()*.
a1 = InputBox("Unesi prvi broj:")	Корисник уноси вредност променљивој а преко параметра *a1*.
b1 = InputBox("Unesi drugi broj:")	Корисник уноси вредност променљивој *b* преко параметра *b1*.
End Sub	Крај процедуре *unesiBrojeve()*.

Sub **prikaziBrojeve**(ByVal a1 As Integer, ByVal b1 As Integer)	Поч. процедуре *prikaziBrojeve().*
MsgBox ("Prvi broj je: " & a1 & " Drugi broj je:" & b1)	Приказ вредности параметара *a* и *b*.
End Sub	Крај процедуре *prikaziBrojeve()*

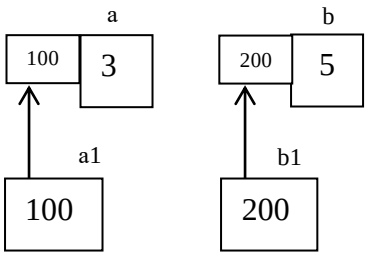
Када се из главне процедуре позове процедура *unesiBrojeve(),* стварни параметри *a* и *b* (који су уједно и промењљиве) се преносе преко адресе. То значи да ће формални параметри *a1* и *b1* добити адресе од промењљивих *a* и *b.*



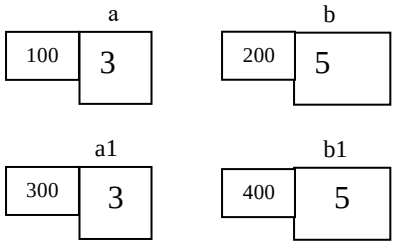
Након извршења наредби:

a1 = InputBox("Unesi prvi broj:")
b1 = InputBox("Unesi drugi broj:")

промењљиве a и b добијају вредности од крајњег корисника преко функције *InputBox(),* посредно помоћу параметара a1 и b1:



Након тога се контрола програма враћа до главне процедуре која позива процедуру *PrikaziBrojeve().* Стварни параметри *a* и *b* (који су уједно и промењљиве) се преносе преко вредности. То значи да ће формални параметри *a1* и *b1* постати посебне копије (на посебној меморијској локацији) које ће добити вредности од промењљивих *a* и *b:*



Када се изврши наредба:

MsgBox ("Prvi broj je: " & a1 & " Drugi broj je:" & b1)

на излазу се приказују вредности параметара *a1* и *b1.*

Кориснички захтев ПР2: *Унети два броја преко улазне функције InputBox(). Сабрати та два броја и приказати их преко функције MsgBox(). Програм урадити преко процедура.*

Private **Sub Comand1_Click**()	
Dim a As Integer	
Dim b As Integer	
Dim c As Integer	
unesiBrojeve a, b	
saberiBrojeve a, b, c	позив процедуре *saberiBrojeve()* којој се прослеђују вредности променљивих *a* и *b* (пренос преко вредности) и адреса променљиве *c* (пренос преко адресе).
prikaziBroj a, b, c	
End Sub	
Sub **unesiBrojeve**(ByRef a1 As Integer, ByRef b1 As Integer)	
a1 = InputBox("Unesi prvi broj:")	
b1 = InputBox("Unesi drugi broj:")	
End Sub	
Sub **saberiBrojeve**(ByVal a2 As Integer, ByVal b2 As Integer, ByRef c2 As Integer)	
c2 = a2 + b2	
End Sub	
Sub **prikaziBroj**(ByVal a3 As Integer, ByVal b3 As Integer, ByVal c3 As Integer)	
MsgBox "Zbir brojeva: " & a3 & " i " & b3 & " je: " & c3	
End Sub	

Задатак ЗП1*: Нацртати изглед меморије за наведени пример ПР2.*

Кориснички захтев ПР3: *Унети број преко улазне функције InputBox(). Одредити да ли је број паран или непаран. Приказати преко MsgBox() функције поруку о томе. Програм урадити преко процедура.*

```
Private Sub Comand2_Click()
Dim a As Integer

unesiBroj a
PrikaziPoruku a
End Sub

Sub unesiBroj(ByRef a1 As Integer)
a1 = InputBox("Unesi broj:")
End Sub

Sub PrikaziPoruku(ByVal a2 As Integer)
If (a2 Mod 2) = 0 Then
    MsgBox ("Broj: " & a & " je paran.")
Else
    MsgBox ("Broj: " & a & " je neparan.")
End If
End Sub
```

Кориснички захтев ПР4: *Унети низ бројева преко улазне функције InputBox(). Приказати низ бројева преко MsgBox() функције. Програм урадити преко процедура.*

```
Private Sub Comand3_Click()
Dim n(5) As Integer
unesiNizBrojeva n
prikazNizaBrojeva n
End Sub

Sub unesiNizBrojeva(ByRef n() As Integer)
Dim i As Integer
MsgBox ("Unesite niz od 5 brojeva:")
For i = 1 To 5
      n(i) = InputBox("Unesi " & i & ". broj:")
Next i
End Sub

Sub prikazNizaBrojeva(ByRef n() As Integer)
Dim i As Integer
For i = 1 To 5
    MsgBox (i & ". broj je: " & n(i))
Next i
End Sub
```

Кориснички захтев ПР5: *Унети дан у недељи преко улазне функције InputBox(). Приказати редни број дана у недељи преко MsgBox() функције. Програм урадити преко процедура.*

```
Private Sub Comand4_Click()
Dim dan As String
unesiDan dan
prikaziRedniBrojDana dan
End Sub

Sub unesiDan(ByRef dan As String)
dan = InputBox("Unesi dan u nedelji:")
End Sub

Sub prikaziRedniBrojDana(ByVal dan As String)
Select Case dan
Case "ponedeljak"
   MsgBox ("Ponedeljak je prvi dan u nedelji.")
Case "utorak"
   MsgBox ("Utorak je drugi dan u nedelji.")
Case "sreda"
   MsgBox ("Sreda je treci dan u nedelji.")
Case "cetvrtak"
   MsgBox ("Cetvrtak je cetvrti dan u nedelji.")
Case "petak"
   MsgBox ("Petak je peti dan u nedelji.")
Case "subota"
   MsgBox ("Subota je sesti dan u nedelji.")
Case "nedelja"
   MsgBox ("Nedelja je sedmi dan u nedelji.")
Case Else
   MsgBox ("Nije unet ispravan naziv dana u nedelji")
End Select
End Sub
```

Задаци:

1. ***ЗПР1 -*** *Унети број преко улазне функције InputBox. Одредити да ли је број позитиван, негативан или је 0. Приказати преко MsgBox функције поруку о томе. Програм урадити преко процедура.*
   ЗПР2 - *Унети назив једног од летњих месеци преко улазне функције InputBox. Приказати редни број месеца у години преко MsgBox функције. Програм урадити преко процедура.*

2.4 Функције

Функција је процедура која као резултат свога извршења враћа неку вредност. Функција се дефинише на следећи начин:

Function ImeFunkcije(par1 as NekiTipPod,par2 as NekiTipPod,...) as TipRezultataFunkcije

' telo funkcije

End Function

Function представља кључну реч која указује на почетак дефинисања функције. Свака функција има јединствено име (*ImeFunkcije*) у модулу, слично процедури. Иза назива функције дефинише се листа параметара функције (*par1,par2,...*). Листа параметара може да буде празна. Сваки од параметара је везан за неки од типова података. Након параметара се наводи тип онога што функција враћа (*TipRezultataFunkcije*). Затим се у телу функције уносе наредбе које одређују шта ће се десити у функцији. На крају се ставља *End Function* кључне речи које указују на крај функције. Пренос параметара код функција је исти као код процедура. Параметри се преносе преко адресе и преко вредности.

Примери:

1. ***Кориснички захтев*** *ФУ1: Унети два броја преко улазне функције InputBox(). Помножити та два броја и приказати их преко функције MsgBox(). Програм урадити коришћењем функција и процедура.*

Private Sub **Comand6_Click**()	
Dim a As Integer	
Dim b As Integer	
Dim c As Integer	
unesiBrojeve a, b	
c = **proizvodBrojeva**(a, b)	Позив функције *proizvodBrojeva()* којој се прослеђују вредности променљивих *a* и *b* (пренос параметара преко вредности). Вредност која се добија након извршења функције *proizvodBrojeva()* се додељује до променљиве *c*.
prikaziPoruku a, b, c	
End Sub	
Function **proizvodBrojeva**(ByVal a As Integer, ByVal b As Integer) As Integer	Почетак функције.
proizvodBrojeva = a * b	Производ вредности параметара a и b се додељују до функције *proizvodBrojeva()*. На тај начин се функција понаша и као променљива која враћа неку вредност.

End Function	Крај функције.
Sub **prikaziPoruku**(ByVal a As Integer, ByVal b As Integer, ByVal c As Integer)	
MsgBox "Proizvod brojeva: " & a & " i " & b & " je: " & c	
End Sub	

2. ***Кориснички захтев*** *ФУ2: Унети број преко улазне функције InputBox(). Одредити да ли је број позитиван, негативан или је 0. Приказати преко MsgBox() функције поруку о томе. Програм урадити преко процедура и функција.*

Private Sub **Comand7_Click**()	
Dim a As Integer	
Dim poruka As String	
unesiBroj a	
poruka = odrediPoruku(a)	Позив функције *odrediPoruku()* којој се прослеђује вредност промењиве *a* (пренос параметра преко вредности). Вредност која се добија након извршења функције *odrediPoruku()* се додељује до промењиве *poruka*.
prikaziPoruku poruka	
End Sub	
Sub **unesiBroj**(ByRef a As Integer)	
a = InputBox("Unesi broj:")	
End Sub	
Function **odrediPoruku**(ByVal a As Integer) As String	Почетак функције.
If a > 0 then	Уколико је вредност параметра *a* позитиван број,
odrediPoruku = "Broj: " & a & " je pozitivan."	тада се функцији *odrediPoruku()* додељује порука да је број позитиван,
Else	иначе се испитује,
if a < 0 then	да ли је број негативан.
odrediPoruku = "Broj: " & a & " je negativan."	Ако је број негативан, функцији *odrediPoruku()* додељује се порука да је број негативан,

else	иначе се,
odrediPoruku = "Uneti broj je 0."	функцији *odrediPoruku()* додељује порука да је унети број 0.
End If	
End If	Крај функције.
End Function	
Sub **prikaziPoruku**(ByVal poruka As String)	
MsgBox (poruka)	
End Sub	

Задатак:

ЗФУ1: *Унети реч преко улазне функције InputBox(). Одредити да ли је реч палиндром. Приказати преко MsgBox() функције поруку о томе. Програм урадити преко процедура и функција.*

2.5 Низови

Низ представља скуп елемената истог типа који се чувају у узастопним меморијским локацијама. Елементима низа се приступа преко индекса низа. Низ се дефинише на следећи начин:

Dim niz(i) As TipPodatka

Код дефинисања низа се одређује колико ће елемената имати низ (*i*). Тип елемената низа *TipPodatka,* као што је речено, је исти за све елементе. Уколико је нпр. дефинисан низ целих бројева од 5 елемената:

Dim niz(5) As Integer

за такав низ кажемо да је **целобројни низ**. Ако се низ састоји од елемената који су знакови, за њега кажемо да је знаковни низ. На сличан начин, у зависности од типа елемената низа, дајемо назив низу.

Примери:

Кориснички захтев НИ1 *– Напунити низ целих бројева. Пронаћи вредност најмањег елемента низа.*

Private Sub **NajmanjiElement_Click**()	Почетак процедуре.
Dim niz(4) As Integer	Декларисање низа од 4 елемента целобројног типа
Dim min As Integer	Декларисање променљиве где ће бити сачувана најмања вредност.
niz(1) = 5	Први елемент низа добија вредност 5.
niz(2) = 4	Други елемент низа добија вредност 4.
niz(3) = 7	Трећи елемент низа добија вредност 7.
niz(4) = 2	Четврти елемент низа добија вредност 2.
For brojac = 1 To 4	У цикличној алгоритамској структури која има разгранату алгоритамску структуру (3 гране), се проналази вредност најмањег елемента низа.
If brojac = 1 Then	
min = niz(brojac)	
Else	
If min > niz(brojac) Then	
min = niz(brojac)	
End If	
End If	
Next	
MsgBox min	Приказ најмање вредности елемента низа.
End Sub	Крај процедуре.

Нацртаћемо табелу наредби и променљивих како би схватили како се извршава програм:

niz	вредност елемента низа
niz(1)	5
niz(2)	4
niz(3)	7
niz(4)	2

Циклус (пролаз)	brojac	If brojac = 1		min
Први	1	1=1, true	**min = niz(1), min = 5**	5
Други	2	2=1, false	**if min > niz(2), 5 > 4, min = niz(2), min = 4**	4
Трећи	3	3=1, false	**if min > niz(3), 4>7, /**	4
Четврти	4	4=1, false	**if min > niz(4), 4>2, min = niz(4), min = 2**	2

Најмања вредност низа је 2.

Кориснички захтев НИ2 *– Напунити низ целих бројева. Пронаћи да ли постоји елемент низа који има вредност једнаку некој задатој вредности.*

Private Sub **PretragaNizaPoZadElementu_Click**()	Почетак процедуре.
Dim niz(4) As Integer	Декларисање низа од 4 елемента целобројног типа.
Dim zad As Integer	Промењљива целобројног типа у којој се чува задата вредност.
Dim signal As Boolean	Декларисање логичке промењљиве.
signal = False	Логичка промењљива је постављена на false. Полази се од предпоставке да ни један елемент низа нема задату вредност.
zad = 7	Задаје се вредност 7.
niz(1) = 5	Пуне се елементи низа са вредностима.
niz(2) = 4	
niz(3) = 7	
niz(4) = 2	
For brojac = 1 To 4	У цикличној алгоритамској структури која има разгранату алг. структуру (2 гране),
If zad = niz(brojac) Then	се проверава да ли постоји елемент у низу који има задату вредност.
signal = True	Постоји елемент који има задату вредност.
End If	
Next	
If signal = True Then	Уколико постоји елемент низа који има задату вредност,
MsgBox " Postoji element sa zadatom vrednošću!"	приказује се порука да постоји елемент са задатом вредношћу,
Else	иначе се приказује порука,
MsgBox " Ne postoji element sa zadatom vrednošću!"	да не постоји елемент са задатом вредношћу,
End If	
End Sub	Крај процедуре.

Нацртаћемо табелу наредби и промењљивих како би схватили како се извршава програм:

niz	вредност елемента низа
niz(1)	5
niz(2)	4
niz(3)	7
niz(4)	2

zad
7

signal
false, true

Циклус (пролаз)	brojac	If zad = niz(brojac)
Први	1	7=niz(1), 7=5, false
Други	2	7=niz(2), 7=4, false
Трећи	3	7=niz(3), 7=7, true, signal = true
Четврти	4	7=niz(4), 7=2, false

Постоји елемент низа који има задату вредност 7.

Кориснички захтев НИЗ *– Напунити низ целих бројева. Приказати вредност која се јавља код два елемента у низу (са понављањем те вредности).*

```
Private Sub DvaPutaUNizu_Click()
Dim niz(4) As Integer
Dim brojac As Integer
niz(1) = 5
niz(2) = 4
niz(3) = 7
niz(4) = 4
‘ Двострука циклична алг. структура, код
‘ које и у унутрашњем и у спољашњем.
‘ циклусу имамо разгранату алг.
‘ структуру са две гране.
For i = 1 To 4                          Почетак спољашње цикличне алг. структуре. Тражи се вредност која се јавља код два елемента низа.
brojac = 0
    For j = 1 To 4                      Почетак унутрашње цикличне алгоритамске структуре.
        If niz(i) = niz(j) Then
            brojac = brojac + 1
        End If
    Next                                Крај унутрашње цикличне алг. структуре.
If brojac = 2 Then
   MsgBox "Vrednost:" & niz(i) & " javlja se
   kod dva elementa."
End If
Next                                    Крај спољашње цикличне алг. структуре.
End Sub
```

Нацртаћемо табелу наредби и променљивих како би схватили како се извршава програм:

niz	вредност елемента низа
niz(1)	5
niz(2)	4
niz(3)	7
niz(4)	4

i	j	If niz(i) = niz(j)	brojac
1	1	If niz(1) = niz(1), 5=5, true, brojac = brojac + 1, brojac = 0 +1, brojac =1	0,1
1	2	If niz(1) = niz(2), 5=4, false	1
1	3	If niz(1) = niz(3), 5=7, false	1
1	4	If niz(1) = niz(4), 5=4, false	1
1	4	If brojac = 2, 1=2, false	1
2	1	If niz(2) = niz(1), 4=5, false	0
2	2	If niz(2) = niz(2), 4=4, true, brojac = brojac + 1, brojac = 0 +1, brojac =1	0,1
2	3	If niz(2) = niz(3), 4=7, false	1
2	4	If niz(2) = niz(4), 4=4, true, brojac = brojac + 1, brojac = 1 +1, brojac =2	1,2
2	4	If brojac = 2, 2=2, true, MsgBox "Vrednost:" & niz(2) & " javlja se kod dva elementa", Vrednost 4 javlja se kod dva elementa.	2
3	1	If niz(3) = niz(1), 7=5, false	0
3	2	If niz(3) = niz(2), 7=4, false	0
3	3	If niz(3) = niz(3), 7=7, true, brojac = brojac + 1, brojac = 0 +1, brojac =1	1
3	4	If niz(3) = niz(4), 7=4, false	1
3	4	If brojac = 2, 1=2, false	1
4	1	If niz(4) = niz(1), 4=5, false	0
4	2	If niz(4) = niz(2), 4=4, true, brojac = brojac + 1, brojac = 0 +1, brojac =1	0,1
4	3	If niz(4) = niz(3), 4=7, false	1
4	4	If niz(4) = niz(4), 4=4, true, brojac = brojac + 1, brojac = 1 +1, brojac =2	1,2
4	4	If brojac = 2, 2=2, true, MsgBox "Vrednost:" & niz(2) & " javlja se kod dva elementa", Vrednost 4 javlja se kod dva elementa.	2

Задатак:

ЗНИ1: *Напунити низ целих бројева. Приказати вредности које се јављају код два елемента у низу (без понављања те вредности). Задатак урадити преко процедура и/или функција.*
Пример:
За улазни низ 1 2 1 2 3 програм треба да прикаже 1 и 2.

Кориснички захтев НИ4 *– Напунити низ целих бројева. Израчунати и приказати суму вредности елемената низа.*

```
Private Sub SumaElemenataNiza_Click()
Dim niz(4) As Integer
Dim suma As Integer
niz(1) = 5
niz(2) = 4
niz(3) = 7
niz(4) = 2
suma = 0
For i = 1 To 4
    suma = suma + niz(i)
Next
MsgBox "Suma vrednosti elemenata je:" & suma
End Sub
```

Нацртаћемо табелу наредби и промењивих како би схватили како се извршава програм:

niz	**вредност елемента низа**
niz(1)	5
niz(2)	4
niz(3)	7
niz(4)	2

i	**suma = suma + niz(i)**
1	suma = suma + niz(1), suma = 0 + 5, suma = 5
2	suma = suma + niz(2), suma = 5 + 4, suma = 9
3	suma = suma + niz(3), suma = 9 + 7, suma = 16
4	suma = suma + niz(4), suma = 16 + 2, suma = 18

Сума вредности елемената низа је 18.

Кориснички захтев НИ5 *– Напунити низ целих бројева. Приказати позитивне вредности елемената низа.*

```
Private Sub PozitivniElementi_Click()
Dim niz(4) As Integer
niz(1) = 5
niz(2) = -4
niz(3) = 7
niz(4) = -2
For i = 1 To 4
  If niz(i) > 0 Then
      MsgBox "Vrednost:" & niz(i) & " je pozitivna."
  Else
      MsgBox „Vrednost:“ & niz(i) & „ nije pozitivna.“
  End If
Next
End Sub
```

Нацртаћемо табелу наредби и променљивих како би схватили како се извршава програм:

niz	вредност елемента низа
niz(1)	5
niz(2)	-4
niz(3)	7
niz(4)	-2

i	If niz(i) > 0
1	if niz(1) > 0, 5 >0, true, MsgBox "Vrednost:" & niz(i) & " je pozitivna.", Vrednost 5 je pozitivna.
2	if niz(2) > 0, -4 >0, false, MsgBox "Vrednost:" & niz(i) & " nije pozitivna.", Vrednost -4 nije pozitivna.
3	if niz(3) > 0, 7 >0, true, MsgBox "Vrednost:" & niz(i) & " je pozitivna.", Vrednost 7 je pozitivna.
4	if niz(4) > 0, -2 >0, false, MsgBox "Vrednost:" & niz(i) & " nije pozitivna.", Vrednost -2 nije pozitivna.

Кориснички захтев НИ6 *– Напунити низ целих бројева. Сортирати низ методом мехурова. На крају приказати сортирани низ.*

```
Private Sub SortiranjeNiza_Click()
Dim niz(4) As Integer
Dim pom As Integer
niz(1) = 5
niz(2) = 4
niz(3) = 7
niz(4) = 2
For i = 1 To 3
   For j = i + 1 To 4
      If niz(i) > niz(j) Then
         pom = niz(i)
         niz(i) = niz(j)
         niz(j) = pom
      End If
   Next
Next
For i = 1 To 4
   MsgBox niz(i)
Next
End Sub
```

Задаци:

ЗНИ2: *Нацртати табелу наредби и променљивих за пример3Н6.*

ЗНИ3: *Написати програм који рачуна разлику два низа (A = B razlika C).*

Кориснички захтев НИ7 *– Напунити низ A са унијом елемената низова B и C (без понављања елемената). Пример:* A = B U C B = {5,4,7,2} C = {7,8,9,4} A = {5,4,7,2,8,9}

```
Private Sub UnijaNizova_Click()
Dim A() As Integer
Dim B(4) As Integer
Dim C(4) As Integer
Dim brojac As Integer
brojac = 0
napuniNiz B, "B"
napuniNiz C, "C"
A = nadjiUnijuNizova(B, C, brojac)
prikaziNiz A, brojac, "A"

End Sub

Sub napuniNiz(ByRef niz() As Integer, nazivNiza As String)
  MsgBox "Uneti niz " & nazivNiza & ":"
   For i = 1 To 4
     niz(i) = InputBox("Uneti " & i & ". element niza:")
   Next i
End Sub

Function nadjiUnijuNizova(B() As Integer, C() As Integer, ByRef brojac As Integer) As Integer()
Dim A(8) As Integer
brojac = 0
For i = 1 To 4
signal = False ' Element niza B se ne nalazi u nizu A
   For j = 1 To brojac
       If B(i) = A(j) Then
         signal = True
       End If
   Next

If (signal = False) Then
   brojac = brojac + 1
   A(brojac) = B(i)
End If

signal = False ' Element niza C se ne nalazi u nizu A
   For j = 1 To brojac
       If C(i) = A(j) Then
         signal = True
       End If
   Next 'Унутрашњи циклус

If (signal = False) Then
   brojac = brojac + 1
   A(brojac) = C(i)
End If

Next ' Спољашњи циклус

nadjiUnijuNizova = A

End Function

Sub prikaziNiz(niz() As Integer, brojac As Integer, nazivNiza As String)
MsgBox "Niz " & nazivNiza & ":"
For i = 1 To brojac
   MsgBox niz(i)
Next
End Sub
```

Кориснички захтев НИ8: *Одредити да ли је улазни низ бројева палиндром.*
Пример 1: 12321 ... наведени улазни низ бројева је палиндром
Пример 2: 123321 ... наведени улазни низ бројева је палиндром

```
Private Sub Palindrom_Click()
Dim A() As Integer
Dim brojElemenataNiza As Integer
Dim signal As Boolean

napuniNiz A, "A", brojElemenataNiza

signal = daLiJeNizPalindrom(A, brojElemenataNiza)

If signal Then
  MsgBox "Uneti niz je palindrom"
Else
  MsgBox "Uneti niz nije palindrom."
End If

End Sub

Sub napuniNiz(ByRef niz() As Integer, nazivNiza As String, ByRef brojElemenataNiza As Integer)
  brojElemenataNiza = InputBox("Uneti broj elemenata niza:")
  ReDim niz(brojElemenataNiza)
  MsgBox "Uneti niz " & nazivNiza & ":"
   For i = 1 To brojElemenataNiza
    niz(i) = InputBox("Uneti " & i & ". element niza:")
   Next i
End Sub

Function daLiJeNizPalindrom(A() As Integer, brojElemenataNiza As Integer) As Boolean
daLiJeNizPalindrom = True
For i = 1 To brojElemenataNiza / 2
  If (A(i) <> A(brojElemenataNiza - i + 1)) Then
     daLiJeNizPalindrom = False
  End If
Next i

End Function
```

Кориснички захтев НИ9: *Направити програм који ће креирати и приказати на стандардном излазу нови низ, за произвољан број елемената низа (n), који ће пратити логику следећег низа: 2,5,8,11,14,...*
Наведени низ је аритметички низ.

```
Private Sub AritmetickiNiz_Click()
Dim niz() As Integer
Dim brojElemenataNiza As Integer

brojElemenataNiza = 0

niz = kreirajNoviNiz(brojElemenataNiza)
prikaziNiz niz, brojElemenataNiza

End Sub


Function kreirajNoviNiz(ByRef brojElemenataNiza As Integer) As Integer()
Dim niz() As Integer

brojElemenataNiza = InputBox("Unesi broj elemenata niza:")
ReDim niz(brojElemenataNiza)

For i = 1 To brojElemenataNiza
    niz(i) = 2 + 3 * (i - 1)
Next i

kreirajNoviNiz = niz
End Function

Sub prikaziNiz(niz() As Integer, brojElemenataNiza As Integer)
  MsgBox "Elementi niza:"
  For i = 1 To brojElemenataNiza
    MsgBox niz(i)
  Next i
End Sub
```

Кориснички захтев НИ10: *Направити програм који ће креирати и приказати на стандардном излазу нови низ, за произвољан број елемената низа (n), који ће пратити логику следећег низа: 2,6,18,54,...*
Наведени низ је геометријски низ.

```
Private Sub GeometrijskiNiz_Click()
Dim niz() As Integer
Dim brojElemenataNiza As Integer

brojElemenataNiza = 0

niz = kreirajNoviNiz(brojElemenataNiza)
prikaziNiz niz, brojElemenataNiza

End Sub


Function kreirajNoviNiz(ByRef brojElemenataNiza As Integer) As Integer()
Dim niz() As Integer

brojElemenataNiza = InputBox("Unesi broj elemenata niza:")
ReDim niz(brojElemenataNiza)

For i = 1 To brojElemenataNiza
    niz(i) = 2 * 3 ^ (i - 1)
Next i

kreirajNoviNiz = niz
End Function

Sub prikaziNiz(niz() As Integer, brojElemenataNiza As Integer)
  MsgBox "Elementi niza:"
  For i = 1 To brojElemenataNiza
    MsgBox niz(i)
  Next i
End Sub
```

Задаци:

ЗНИ4: *Направити програм који ће креирати и приказати на стандардном излазу нови низ, за произвољан број елемената низа (n), који ће пратити логику следећег низа: 1,3,5,7,...*
Наведени низ је аритметички низ.

2.6 Матрице

Матрица представља скуп елемената истога типа који су распоређени у табели која има одређен број редова и колона. Елементима матрице се приступа преко индекса (i,j) матрице, где *i* указује на ред а *j* на колону матрице. Матрица се дефинише на следећи начин:

```
Dim mat(m, n) As TipPodatka
```

Код дефинисања матрице се одређује колико ће редова (*m*) и колона (*n*) имати матрица. Тип елемената матрице *TipPodatka,* као што је речено, је исти за све елементе матрице. Уколико је нпр. дефинисана следећа матрица:

```
Dim mat(4,3) As Integer
```

за такву матрицу кажемо да је **целобројна матрица,** јер су њени елементи целобројног типа. Низ је специјалан случај матрице која има један ред и више колона. За матрицу се често каже да она представља низ низова.

Примери

Кориснички захтев MA1 *– Напунити и приказати вредности елемената целобројне матрице са 2 реда и 3 колоне.*

Private Sub **UnosiPrikazMatrice_Click**()	Почетак процедуре.
Dim mat(2, 3) As Integer	Декларисање матрице од 2 реда и 3 колоне која има елементе целобројног типа
UnosMatrice mat, 2, 3	Позив процедуре помоћу које се пуни матрица, матрица се прослеђује као параметар. Пренос параметра матрице је преко адресе. Остали параметри (број редова и колона) се прослеђују преко вредности.
PrikazMatrice mat, 2, 3	Позив процедуре која приказује вредности елемената матрице
End Sub	
Sub **UnosMatrice**(ByRef mat() As Integer, ByVal brojRedova As Integer, ByVal brojKolona As Integer)	
Dim i, j As Integer	
For i = 1 To brojRedova	
For j = 1 To brojKolona	
mat(i, j) = InputBox("Unesi x[" & i & "][" & j & "] element matrice:")	Корисник уноси вредности елемената матрице коришћењем функције *InputBox().*
Next j	
Next i	
End Sub	

Sub **PrikazMatrice**(ByRef mat() As Integer, ByVal brojRedova As Integer, ByVal brojKolona As Integer)	
Dim i, j As Integer	
For i = 1 To brojRedova	
For j = 1 To brojKolona	
MsgBox "x[" & i & "][" & j & "] element matrice:" & mat(i, j)	Приказ елемената матрице на излазу
Next j	
Next i	
End Sub	

Нацртаћемо табелу наредби и променљивих како би схватили како се извршава програм:

m (2,3)	j=1	j=2	j=3
i=1	5	4	2
i=2	7	1	6

Извршење процедуре *UnosMatrice()*.

I	j	mat(i, j) = InputBox("Unesi x[" & i & "][" & j & "] element matrice:")
1	1	mat(1, 1) = InputBox("Unesi x[1][1] element matrice:"), mat(1,1) = 5
1	2	mat(1, 2) = InputBox("Unesi x[1][2] element matrice:"), mat(1,2) = 4
1	3	mat(1, 3) = InputBox("Unesi x[1][3] element matrice:"), mat(1,3) = 2
2	1	mat(2, 1) = InputBox("Unesi x[2][1] element matrice:"), mat(2,1) = 7
2	2	mat(2, 2) = InputBox("Unesi x[2][2 element matrice:"), mat(2,2) = 1
2	3	mat(2, 3) = InputBox("Unesi x[2][3] element matrice:"), mat(2,3) = 6

Извршење процедуре *PrikazMatrice()*.

i	j	MsgBox "x[" & i & "][" & j & "] element matrice:" & mat(i, j)
1	1	MsgBox "x[1][1] element matrice: 5" , izlaz: x[1][1] element matrice: 5
1	2	MsgBox "x[1][2] element matrice: 4" , izlaz: x[1][2] element matrice: 4
1	3	MsgBox "x[1][3] element matrice: 2" , izlaz: x[1][3] element matrice: 2
2	1	MsgBox "x[2][1] element matrice: 7" , izlaz: x[2][1] element matrice: 7
2	2	MsgBox "x[2][2] element matrice: 1" , izlaz: x[2][2] element matrice: 1
2	3	MsgBox "x[2][3] element matrice: 6" , izlaz: x[2][3] element matrice: 6

Кориснички захтев MA2 *– Напунити и сабрати две целобројне матрице које имају 2 реда и 3 колоне.*

Private Sub **SabiranjeDveMatrice_Click**()	
Dim mat1(2, 3) As Integer	Декларисање три целобројне матрице од 2 реда и 3 колоне, матрице *mat1* и *mat2* се сабирају и резултат се чува у матрици *mat3*.
Dim mat2(2, 3) As Integer	
Dim mat3(2, 3) As Integer	
MsgBox "Unos prve matrice:"	
UnosMatrice mat1, 2, 3	
MsgBox "Unos druge matrice:"	
UnosMatrice mat2, 2, 3	
SabiranjeMatrice mat1, mat2, mat3, 2, 3	Позив процедуре која сабира матрице.
PrikazMatrice mat3, 2, 3	
End Sub	
Sub **UnosMatrice**(ByRef mat() As Integer, ByVal brojRedova As Integer, ByVal brojKolona As Integer)	
Dim i, j, a As Integer	
For i = 1 To brojRedova	
For j = 1 To brojKolona	
mat(i, j) = InputBox("Unesi x[" & i & "][" & j & "] element matrice:")	
Next j	
Next i	
End Sub	
Sub **SabiranjeMatrice**(ByRef mat1() As Integer, ByRef mat2() As Integer, ByRef mat3() As Integer, ByVal brojRedova As Integer, ByVal brojKolona As Integer)	
For i = 1 To brojRedova	
For j = 1 To brojKolona	
mat3(i, j) = mat1(i, j) + mat2(i, j)	
Next j	
Next i	
End Sub	

```
Sub PrikazMatrice(ByRef mat() As Integer, ByVal brojRedova As Integer, ByVal brojKolona As Integer)
Dim i, j As Integer
For i = 1 To brojRedova
  For j = 1 To brojKolona
      MsgBox "x[" & i & "][" & j & "] element matrice:" & mat(i, j)
  Next j
Next i
End Sub
```

Нацртаћемо табелу наредби и променљивих како би схватили како се извршава програм:

mat1 (2,3)	j=1	j=2	j=3
i=1	5	4	2
i=2	7	1	6

mat2 (2,3)	j=1	j=2	j=3
i=1	3	1	2
i=2	5	4	3

mat3 (2,3)	j=1	j=2	j=3
i=1	8	5	4
i=2	12	5	9

Извршење процедуре *SabiranjeMatrice()*.

i	j	mat3(i, j) = mat1(i, j) + mat2(i, j)
1	1	mat3(1, 1) = mat1(1, 1) + mat2(1, 1), mat3(1,1) = 5 + 3, mat3(1,1) = 8
1	2	mat3(1, 2) = mat1(1, 2) + mat2(1, 2), mat3(1,2) = 4 + 1, mat3(1,2) = 5
1	3	mat3(1, 3) = mat1(1, 3) + mat2(1, 3), mat3(1,3) = 2 + 2, mat3(1,3) = 4
2	1	mat3(2, 1) = mat1(2, 1) + mat2(2, 1), mat3(2,1) = 7 + 5, mat3(2,1) = 12
2	2	mat3(2, 2) = mat1(2, 2) + mat2(2, 2), mat3(2,2) = 1 + 4, mat3(2,2) = 5
2	3	mat3(2, 3) = mat1(2, 3) + mat2(2, 3), mat3(2,3) = 6 + 3, mat3(2,3) = 9

Кориснички захтев МА3 *– Напунити квадратну целобројну матрицу и сортирати главну дијагоналу матрице.*

```
Private Sub SortiranjeGlavneDijagonaleMatrice_Click()
Dim mat(4, 4) As Integer
UnosMatrice mat, 4, 4
SortiranjeMatrice mat, 4, 4        позив процедуре која сортира матрицу.
PrikazMatrice mat, 4, 4
End Sub

Sub UnosMatrice(ByRef mat() As Integer, ByVal brojRedova As Integer, ByVal brojKolona As Integer)
Dim i, j, a As Integer
For i = 1 To brojRedova
  For j = 1 To brojKolona
      mat(i, j) = InputBox("Unesi x[" & i & "][" & j & "] element matrice:")
  Next j
Next i
End Sub

Sub SortiranjeMatrice(ByRef mat() As Integer, ByVal brojRedova As Integer, ByVal brojKolona As Integer)
Dim i, j, pom As Integer
For i = 1 To brojRedova – 1
   For j = i + 1 To brojRedova
      If (mat(i, i) > mat(j, j)) Then
         pom = mat(i, i)
         mat(i, i) = mat(j, j)
         mat(j, j) = pom
      End If
   Next j
Next i
End Sub

Sub PrikazMatrice(ByRef mat() As Integer, ByVal brojRedova As Integer, ByVal brojKolona As Integer)
Dim i, j As Integer
For i = 1 To brojRedova
   For j = 1 To brojKolona
      MsgBox "x[" & i & "][" & j & "] element matrice:" & mat(i, j)
   Next j
Next i
End Sub
```

Нацртаћемо табелу наредби и променљивих како би схватили како се извршава програм:

mat (3,3)	**j=1**	**j=2**	**j=3**	**j=4**
i=1	**5,1**	4	2	9
i=2	7	**1,5,3,2**	6	1
i=3	4	3	**3,5,3**	3
i=4	2	8	2	**2,3,5**

Извршење процедуре *SortiranjeMatrice()*.

i	j	If (mat(i, i) > mat(j, j))	pom = mat(i, i), mat(i, i) = mat(j, j), mat(j, j) = pom	mat (3,3)	j=1	j=2	j=3	j=4
				i=1	**5**	4	2	9
				i=2	7	**1**	6	1
				i=3	4	3	**3**	3
				i=4	2	8	2	**2**
1	2	If (mat(1, 1) > mat(2, 2)), 5>1, true	pom = mat(1, 1), mat(1, 1) = mat(2, 2), mat(2, 2) = pom, pom = 5, mat(1,1) = 1, mat(2,2) = 5					
				i=1	**1**	4	2	9
				i=2	7	**5**	6	1
				i=3	4	3	**3**	3
				i=4	2	8	2	**2**
1	3	If (mat(1, 1) > mat(3, 3)), 1>3, false	/					
1	4	If (mat(1, 1) > mat(4, 4)), 1>2, false	/					
2	3	If (mat(2, 2) > mat(3, 3)), 5>3, true	pom = mat(2, 2), mat(2, 2) = mat(3, 3), mat(3, 3) = pom, pom = 5, mat(2,2) = 3, mat(3,3) = 5					
				i=1	**1**	4	2	9
				i=2	7	**3**	6	1
				i=3	4	3	**5**	3
				i=4	2	8	2	**2**
2	4	If (mat(2, 2) > mat(4, 4)), 3>2, true	pom = mat(2, 2), mat(2, 2) = mat(4, 4), mat(4, 4) = pom, pom = 3, mat(2,2) = 2, mat(4,4) = 3					
				i=1	**1**	4	2	9
				i=2	7	**2**	6	1
				i=3	4	3	**5**	3
				i=4	2	8	2	**3**
3	4	If (mat(3, 3) > mat(4, 4)), 5>3, false	pom = mat(3, 3), mat(3, 3) = mat(4, 4), mat(4, 4) = pom, pom = 5, mat(3,3) = 3, mat(4,4) = 5					
				i=1	**1**	4	2	9
				i=2	7	**2**	6	1
				i=3	4	3	**3**	3
				i=4	2	8	2	**5**

Изглед матрице m након сортирања главне дијагонале матрице:

mat (3,3)	j=1	j=2	j=3	j=4
i=1	**1**	4	2	9
i=2	7	**2**	6	1
i=3	4	3	**3**	3
i=4	2	8	2	**5**

Кориснички захтев МА4 *– Напунити матрицу која има 3 реда и 4 колоне и матрицу (вектор) која има 4 реда и 1 колону. Помножити матрицу и вектор и приказати резултат.*

Напомена: Када се множи матрица m са вектором v, број колона матрице мора да буде једнак броју редова вектора. Резултат множења је вектор v1 који има а) број редова једнак броју редова матрице m и б) број колона који је једнак броју колона (једна колона) вектора v:

v1(r1,k2) = m(r1,k1) * v(r2,k2), где је k2 = 1

На пример ако се помножи матрица m која има три реда и 4 колоне,

mat[3,4]

2	1	3	4
4	2	4	1
6	3	7	2

са вектором v који има 4 реда и 1 колону,

v[4,1]

2
1
3
5

резултат је вектор v1 који има три реда и једну колону.

v1[3,1]

34
27
46

Private Sub **MnozenjeMatriceIVektora_Click**()	
Dim mat1(3, 4) As Integer	
Dim mat2(4, 1) As Integer	
Dim mat3(3, 1) As Integer	
MsgBox "Unos prve matrice:"	
UnosMatrice mat1, 3, 4	
MsgBox "Unos vektora:"	
UnosMatrice mat2, 4, 1	
MnozenjeMatrice mat1, mat2, mat3, 3, 4	позив процедуре која множи матрицу са вектором.
PrikazMatrice mat3, 3, 1	
End Sub	

```
Sub UnosMatrice(ByRef mat() As Integer, ByVal brojRedova As Integer, ByVal brojKolona As Integer)
Dim i, j, a As Integer
For i = 1 To brojRedova
  For j = 1 To brojKolona
      mat(i, j) = InputBox("Unesi x[" & i & "][" & j & "] element matrice:")
  Next j
Next i
End Sub

Sub MnozenjeMatrice(ByRef mat1() As Integer, ByRef mat2() As Integer, ByRef mat3() As Integer, ByVal brojRedova As Integer, ByVal brojKolona As Integer)
For i = 1 To brojRedova
  mat3(i, 1) = 0
  For j = 1 To brojKolona
      mat3(i, 1) = mat3(i, 1) + (mat1(i, j) * mat2(j, 1))
  Next j
Next i
End Sub

Sub PrikazMatrice(ByRef mat() As Integer, ByRef  brojRedova As Integer, ByVal brojKolona As Integer)
Dim i, j As Integer
For i = 1 To brojRedova
  For j = 1 To brojKolona
      MsgBox "x[" & i & "][" & j & "] element matrice:" & mat(i, j)
  Next j
Next i
End Sub
```

Нацртаћемо табелу наредби и променљивих како би схватили како се извршава програм:

mat1 (3,4)	j=1	j=2	j=3	j=4
i=1	2	1	3	4
i=2	4	2	4	1
i=3	6	3	7	2

mat2 (4,1)	j=1
i=1	2
i=2	1
i=3	3
i=4	5

mat3 (3,1)	j=1
i=1	0,4,5,14,34
i=2	0,8, 10,22,27
i=3	0,12,15,36,46

Извршење процедуре *MnozenjeMatrice()*.

i	j	mat3(i, 1) = 0	mat3(i, 1) = mat3(i, 1) + (mat1(i, j) * mat2(j, 1)) - aa
1	1	mat3(1, 1) = 0, .. **aa**.., mat3(1, 1) = 4	mat3(1, 1)=mat3(1, 1) + (mat1(1, 1) * mat2(1, 1)) mat3(1, 1) = 0 + (2*2), mat3(1,1) = 4
1	2	mat3(1, 1) = 4, .. **aa**.., mat3(1, 1) = 5	mat3(1, 1)=mat3(1, 1) + (mat1(1, 2) * mat2(2, 1)) mat3(1, 1) = 4 + (1*1), mat3(1,1) = 5
1	3	mat3(1, 1) = 5, .. **aa**.., mat3(1, 1) = 14	mat3(1, 1)=mat3(1, 1) + (mat1(1, 3) * mat2(3, 1)) mat3(1, 1) = 5 + (3*3), mat3(1,1) = 14
1	4	mat3(1, 1) = 14, .. **aa** .. , mat3(1, 1) = 34	mat3(1, 1)=mat3(1, 1) + (mat1(1, 4) * mat2(4, 1)) mat3(1, 1) = 14 + (4*5), mat3(1,1) = 34
2	1	mat3(2, 1) = 0, .. **aa**.., mat3(2, 1) = 8	mat3(2, 1)=mat3(2, 1) + (mat1(2, 1) * mat2(1, 1)) mat3(2, 1) = 0 + (4*2), mat3(2,1) = 8
2	2	mat3(2, 1) = 8, .. **aa**.., mat3(2, 1) = 10	mat3(2, 1)=mat3(2, 1) + (mat1(2, 2) * mat2(2, 1)) mat3(2, 1) = 8 + (2*1), mat3(2,1) = 10
2	3	mat3(2, 1) = 10, .. **aa**.., mat3(2, 1) = 22	mat3(2, 1)=mat3(2, 1) + (mat1(2, 3) * mat2(3, 1)) mat3(2, 1) = 10 + (4*3), mat3(2,1) = 22
2	4	mat3(2, 1) = 22, .. **aa**.., mat3(2, 1) = 27	mat3(2, 1)=mat3(2, 1) + (mat1(2, 4) * mat2(4, 1)) mat3(2, 1) = 22 + (1*5), mat3(2,1) = 27
3	1	mat3(3, 1) = 0, .. **aa**.., mat3(2, 1) = 12	mat3(3, 1)=mat3(3, 1) + (mat1(3, 1) * mat2(1, 1)) mat3(3, 1) = 0 + (6*2), mat3(3,1) = 12
3	2	mat3(3, 1) = 12, .. **aa**.., mat3(2, 1) = 15	mat3(3, 1)=mat3(3, 1) + (mat1(3, 2) * mat2(2, 1)) mat3(3, 1) = 12 + (3*1), mat3(3,1) = 15
3	3	mat3(3, 1) = 15, .. **aa**.., mat3(2, 1) = 36	mat3(3, 1)=mat3(3, 1) + (mat1(3, 3) * mat2(3, 1)) mat3(3, 1) = 15 + (7*3), mat3(3,1) = 36
3	4	mat3(3, 1) = 36, .. **aa**.., mat3(2, 1) = 46	mat3(3, 1)=mat3(3, 1) + (mat1(3, 4) * mat2(4, 1)) mat3(3, 1) = 36 + (2*5), mat3(3,1) = 46

Кориснички захтев МА5 *– Напунити матрицу која има 2 реда и 3 колоне и матрицу која има 3 реда и 4 колоне. Помножити матрице и приказати резултат.*

Напомена: Када се множи матрица m1 са матрицом m2, број колона матрице m1 мора да буде једнак броју редова матрице m2. Резултат множења је матрица m3 који има а) број редова једнак броју редова матрице m1 и б) број колона који је једнак броју колона матрице m2:

m3(r1,k2) = m1(r1,k1) * m2(r2,k2)

На пример ако се помножи матрица m1 која има 2 реда и 3 колоне,

m1[2,3]

2	4	5
1	7	2

са матрицом m2 која има 3 реда и 4 колоне,

m2[3,4]

3	2	1	4
5	3	1	2
1	4	2	3

резултат је матрица m3 који има 2 реда и 4 колоне.

m3[2,4]

31	36	16	31
40	31	12	24

```
Private Sub MnozenjeDveMatrice_Click()
Dim mat1(2, 3) As Integer
Dim mat2(3, 4) As Integer
Dim mat3(2, 4) As Integer
MsgBox "Unos prve matrice:"
UnosMatrice mat1, 2, 3
MsgBox "Unos druge matrice:"
UnosMatrice mat2, 3, 4
MnozenjeMatrica mat1, mat2, mat3, 2, 3, 4
PrikazMatrice mat3, 2, 4
End Sub
Sub UnosMatrice(ByRef mat() As Integer, ByVal brojRedova As Integer, ByVal brojKolona As Integer)
Dim i, j, a As Integer
For i = 1 To brojRedova
  For j = 1 To brojKolona
      mat(i, j) = InputBox("Unesi x[" & i & "][" & j & "] element matrice:")
  Next j
Next i
End Sub
Sub MnozenjeMatrica(ByRef mat1() As Integer, ByRef mat2() As Integer, ByRef mat3() As Integer, ByVal brojRedova As Integer, ByVal brojRedovaKolona As Integer, ByVal brojKolona As Integer)
For i = 1 To brojRedova
  For j = 1 To brojKolona
      mat3(i, j) = 0
      For k = 1 To brojRedovaKolona
          mat3(i, j) = mat3(i, j) + (mat1(i, k) * mat2(k, j))
      Next k
   Next j
Next i
End Sub

Sub PrikazMatrice(ByRef mat() As Integer, ByVal brojRedova As Integer, ByVal brojKolona As Integer)
Dim i, j As Integer
For i = 1 To brojRedova
  For j = 1 To brojKolona
      MsgBox "x[" & i & "][" & j & "] element matrice:" & mat(i, j)
  Next j
Next i
End Sub
```

Задаци:

ЗМА1: *Нацртати табелу наредби и променљивих за пример*

ЗМА2*. Приказати извршење процедуре MnozenjeMatrice1().*

ЗМА3: *Упоредити да ли су две матрице исте.*

2.7 Слогови

Слог можемо посматрати на два начина:
а) Слог као тип (слоговни тип) је општи представник неког скупа елемената. Он се може дефинисати на следећи начин:

```
Type ImeTipa
        Polje1 as TipPodataka1
                Polje2 as TipPodataka2
           ...
                Poljen as TipPodatakan
End Type
```

TipPodataka$_1$, TipPodataka$_2$, ..., TipPodataka$_n$ – типови могу бити исти или различити
n – број поља (атрибута) слоговног типа

На пример уколико желимо да прикажемо слоговни тип *Student,* који има три поља (*brojIndeksa, ImePrezime,GodinaStudija*), то изгледа овако:

```
Type Student
   BrojIndeksa As String
   ImePrezime As String
   GodinaStudija As Integer
End Type
```

б) Слог се може посматрати као један елемент из скупа елемената слоговног типа. У том случају слог називамо променљива слоговног типа. На пример, уколико декларишемо,

```
Dim st As Student
```

st ће бити један елемент из скупа елемената слоговног типа *Student*, односно променљива слоговног типа *Student.* Наведени слог се у меморији може представити на следећи начин:

st

BrojIndeksa	ImePrezime	GodinaStudija

Уколико желимо да приступимо појединим пољима слога, нпр. уколико желимо да пољу *BrojIndeksa* слога *st* додамо неку вредност, то радимо на следећи начин:

```
st.BrojIndeksa= “123-2010”
```

У општем смислу слогу *sl* слоговног типа *ST* приступамо преко квалификатора (.):

sl.Poljei

где је i = (1,n), n је број поља слоговног типа ST

```
Type ST
        Polje1 as TipPodataka1
                ...
                Poljen as TipPodatakan
End Type
```

Примери:

Кориснички захтев СЛ1: *Дефиниши слоговни тип Student који има 3 поља, број индекса, име и презиме и година студија. Направи, напуни и прикажи промењиву слоговног типа.*

Type **Student**	Дефинисање слоговног типа.
BrojIndeksa As String	
ImePrezime As String	
GodinaStudija As Integer	
End Type	
Private Sub **SL1_Click**()	Почетак процедуре.
Dim st As Student	Декларисање слоговне промењиве.
st.BrojIndeksa = InputBox("Unesi broj indeksa:")	Унос вредности поља слога *st*.
st.ImePrezime = InputBox("Unesi ime i prezime:")	
st.GodinaStudija = InputBox("Unesi godinu studija:")	
MsgBox "Broj indeksa: " & st.BrojIndeksa & " Ime i prezime:" & st.ImePrezime & " Godina studija:" & st.GodinaStudija	Приказ слога
End Sub	Крај процедуре.

Нацртаћемо изглед напуњеног објекта st у меморији:

st

BrojIndeksa	ImePrezime	GodinaStudija
123-2010	Пера Перић	4

Кориснички захтев СЛ2: *Дефинишии слоговни тип Student који има 3 поља, број индекса, име и презиме и година студија. Направи, напуни и прикажи* ***низ променљивих*** *слоговног типа.*

Type **Student**	
BrojIndeksa As String	
ImePrezime As String	
GodinaStudija As Integer	
End Type	
Private Sub **SL2_Click**()	Почетак процедуре.
Dim st(3) As Student	Декларисање слоговне низовне променљиве од три елемента (слога).
For i = 1 To 3	
st(i).BrojIndeksa = InputBox("Unesi broj indeksa:")	Унос вредности слоговима низовне променљиве *st.*
st(i).ImePrezime = InputBox("Unesi ime i prezime:")	
st(i).GodinaStudija = InputBox("Unesi godinu studija:")	
Next	
For i = 1 To 3	
MsgBox "Broj indeksa: " & st(i).BrojIndeksa & " Ime i prezime:" & st(i).ImePrezime & " Godina studija:" & st(i).GodinaStudija	Приказ слогова.
Next	
End Sub	Крај процедуре.

Нацртаћемо табелу наредби и променљивих како би схватили како се извршава програм:

st(3)	BrojIndeksa	ImePrezime	GodinaStudija
(1)	"123-2010"	"Pera Peric"	4
(2)	"124-2010"	"Milan Savic"	4
(3)	"125-2010"	"Jovan Babic"	3

Унос вредности слоговима низовне променљиве st.

i	st(i).BrojIndeksa = InputBox("Unesi broj indeksa:")	st(i).ImePrezime = InputBox("Unesi ime i prezime:")	st(i).GodinaStudija = InputBox("Unesi godinu studija:")
1	st(1).BrojIndeksa = "123-2010"	st(1).ImePrezime = "Pera Peric"	st(1).GodinaStudija = 4
2	st(2).BrojIndeksa = "124-2010"	st(2).ImePrezime = "Milan Savic"	st(2).GodinaStudija = 4
3	st(3).BrojIndeksa = "125-2010"	st(3).ImePrezime = "Jovan Babic"	st(3).GodinaStudija = 3

Задатак:

ЗСЛ1: *Написати програм који дефинише слоговни тип Nastavnik која има три атрибута: ImePrezime, Adresa, Predmet. Дефинисати низ од 4 слоговне променљиве типа Nastavnik. Напунити наведени низ. Приказати само оне наставнике који предају предмет "Matematika".*

Кориснички захтев СЛ3: *Дефинишиу слоговни тип* Student *који има 3 поља, број индекса, име и презиме и година студија. Направи, напуни и прикажи* ***низ променљивих*** *слоговног типа* Student*. Након тога сортирај (по броју индекса) и прикажи низ у растућем редоследу.*

```
Private Sub SL3_Click()
Dim st(5) As Student
NapuniNiz st, 5
SortirajNiz st, 5
PrikaziNiz st, 5
End Sub

Private Sub NapuniNiz(ByRef st() As Student, ByVal brojStudenata As Integer)
For i = 1 To brojStudenata
  st(i).BrojIndeksa = InputBox("Unesi broj indeksa:")
  st(i).ImePrezime = InputBox("Unesi ime i prezime:")
  st(i).GodinaStudija = InputBox("Unesi godinu studija:")
Next
End Sub

Private Sub SortirajNiz(ByRef st() As Student, ByVal brojStudenata As Integer)
Dim BrojIndeksa As String
Dim ImePrezime As String
Dim GodinaStudija As Integer
For i = 1 To brojStudenata - 1
   For j = i + 1 To brojStudenata
      If StrComp(st(i).BrojIndeksa, st(j).BrojIndeksa) = 1 Then
         BrojIndeksa = st(i).BrojIndeksa
         ImePrezime = st(i).ImePrezime
         GodinaStudija = st(i).GodinaStudija

         st(i).BrojIndeksa = st(j).BrojIndeksa
         st(i).ImePrezime = st(j).ImePrezime
         st(i).GodinaStudija = st(j).GodinaStudija

         st(j).BrojIndeksa = BrojIndeksa
         st(j).ImePrezime = ImePrezime
         st(j).GodinaStudija = GodinaStudija
      End If
   Next j
Next i
End Sub

Private Sub PrikaziNiz(ByRef st() As Student, ByVal brojStudenata As Integer)
For i = 1 To brojStudenata
MsgBox "Broj indeksa: " & st(i).BrojIndeksa & " Ime i prezime:" &
st(i).ImePrezime & " Godina studija:" & st(i).GodinaStudija
Next
End Sub
```

Задатак:

ЗСЛ2: *За пример ЗСЛ1 нацртати табелу наредби и променљивих како би схватили како се извршава програм.*

ЗСЛ3: *Допунити задатак ЗСЛ1 са процедуром која приказује све студенте изабране године студија.*

ЗСЛ4: *Допунити задатак ЗСЛ1 са процедуром која приказује укупан број студената изабране године студија.*

2.8 Датотеке

Датотечни систем (file system) se састоји од директоријума (фолдера) и датотека које се налазе на спољној меморији (диск, CD, флеш, ...). Директоријум је организациона јединица спољне меморије која има свој назив и која може имати више директоријума и датотека. Датотека је организациона јединица директоријума која има свој назив и која садржи податке. Наведена организација спољне меморије омогућава да се прецизно опише путања до сваке датотеке (ImeSpoljneMemorije\ImeDirektorijuma\ImeDatoteke). Нпр. Ако на диску постоји директоријум Kurs, а у њему поддиректоријум Osnovni у коме се чува датотека tekst.txt, тада ће путања до ове датотеке бити: C:\Kurs\Osnovni\tekst.txt
У следећих неколико примера показаћемо како се креирају датотеке у VB-у, и како се уписују и читају подаци из датотека.

Примери

1. Кориснички захтев DAT1 – *Напуни датотеку са низом бројева, затим прочитај датотеку и прикажи парне бројеве.*

Private Sub **Dat1_Click**()	Почетак процедуре.
Dim niz(4) As Integer	
Dim pom As Integer	
Dim brojElemenata As Integer	
niz(1) = 5	
niz(2) = 4	
niz(3) = 2	
niz(4) = 7	
Set fs = CreateObject("Scripting.FileSystemObject")	Креирање објекта који представља датотечни систем *fs* (file system) .
Set dat = fs.createTextFile("dat1.txt", True)	На спољној меморији се креира текстуална датотека *dat1.txt*.
dat.WriteLine 4	У датотеци се памти број елемената низа.
For i = 1 To 4	
dat.WriteLine niz(i)	У датотеци се памте вредности елемената низа
Next i	
dat.Close	Затварање датотеке.
‘Отвара се датотека dat1.txt у режиму читања. Други ‘параметар функције *OpenTextFile()* има вредност 1, ‘што значи да се датотека може само читати – не може ‘се у њу ништа уписивати; уколико наведени параметар ‘има вредност 2 тада се у датотеку може вршити унос ‘података; трећи параметар функције *OpenTextFile()* ‘има вредност 0, што значи да ће подаци бити у Unicode ‘формату.	
Set dat = fs.OpenTextFile("dat1.txt", 1, 0)	
brojElemenata = dat.ReadLine	Чита се из датотеке број елемената низа.

MsgBox "Parni brojevi:"	
For i = 1 To brojElemenata	
pom = dat.ReadLine	Чита се вредност елемента низа.
If (pom Mod 2 = 0) Then	
MsgBox pom	Приказују се вредности елемената који су парни бројеви
End If	
Next i	
dat.Close	Затварање датотеке.
End Sub	Крај процедуре.

2. Кориснички захтев ДАТ2 – *Напуни датотеку именима. Прочитај датотеку и прикажи имена код којих је прво слово ‘s’.*

```
Private Sub Dat2_Click()
Dim brojImena As Integer
Dim ime As String
Set fs = CreateObject("Scripting.FileSystemObject")
Set dat = fs.createTextFile("dat2.txt", True)
brojImena = InputBox("Unesi broj imena koji zelis da zapamtis:")
dat.WriteLine brojImena
For i = 1 To brojImena
    ime = InputBox("Unesi " & i & ". ime:")
    dat.WriteLine ime
Next i
dat.Close

Set dat = fs.OpenTextFile("dat2.txt", 1, 0)
brojImena = dat.ReadLine
MsgBox "Imena koja pocinju sa 'S':"
For i = 1 To brojImena
   ime = dat.ReadLine
   If Left(ime, 1) = "S" Then
        MsgBox ime
    End If
Next i
dat.Close
End Sub
```

3. Кориснички захтев ДАТ3 – *Напуни датотеке са знаковима. Прочитај датотеку и прикажи знакове који се јављају два пута (са понављањем).*

```
Private Sub Dat3_Click()
Dim brojZnakova As Integer
Dim znak As String
Dim brojac As Integer
Set fs = CreateObject("Scripting.FileSystemObject")
Set dat = fs.createTextFile("dat3.txt", True)
brojZnakova = InputBox("Unesi broj znakova koji zelis da zapamtis:")
dat.WriteLine brojZnakova
For i = 1 To brojZnakova
   znak = InputBox("Unesi " & i & ". znak:")
   dat.WriteLine znak
Next i
dat.Close
Set dat = fs.OpenTextFile("dat3.txt", 1, 0)
brojZnakova = dat.ReadLine
MsgBox "Znakovi koji se javljaju tacno dva puta(sa ponavljanjem):"
For i = 1 To brojZnakova
    znak = dat.ReadLine
    brojac = 0
'Отвара се датотека dat3.txt у режиму читања. 'Датотеци се приступа преко објекта dat1. То значи 'да више објеката (у овом случају dat и dat1) могу да 'обрађују исту датотеку (dat3.txt).
Set dat1 = fs.OpenTextFile("dat3.txt", 1, 0)
    dat1.ReadLine
    For j = 1 To brojZnakova
       znak1 = dat1.ReadLine
       If znak = znak1 Then
          brojac = brojac + 1
       End If
    Next j
    dat1.Close
     If brojac = 2 Then
        MsgBox "Vrednost:" & znak & " javlja se kod dva elementa."
     End If
 Next i
 dat.Close
End Sub
```

Задатак:

1. ***ЗДАТ1*** *– Напуни датотеку са низом бројева. Прочитај датотеку и прикажи суму тих бројева.*
2. ***ЗДАТ2*** *– Напуни датотеке са знаковима. Прочитај датотеку и прикажи колико има самогласника.*

3. Рад са базом података

База података је организована колекција података која се чува и користи од стране више корисника. База података се састоји од колекције података којима се управља помоћу система за управљање базом података. Податак је носилац информације - кодирана чињеница из реалног система. Информација је: а) капацитет повећања знања б) нешто што укида или смањује неодређеност ц) протумачени (интерпретирани) податак. База података се прави у току развоја **информационог**, односно **софтверског система**. Информациони систем је систем у коме се везе између објеката и везе система са околином остварује разменом информација . Софтверски систем је систем који обезбеђује неки скуп функционалности (функција) за крајњег корисника.

3.1 Контекст базе података

Да би се схватила улога базе података потребно је да се схвати окружење, односно контекст у коме се користи база података. База података се прави у току развоја информационог система (ИС), односно софтверског система[15]. ИС се прави како би се олакшао рад и управљање неким **реалним пословним системом**. Пословни систем у најопштијем смислу има своју **структуру** и **понашање**. Структура пословног система се односи на организациону структуру преко које се одређују везе и односи између елемената пословног система, док се понашање пословног система односи на пословне процесе који одређују токове извршења функција пословног система (нпр. функција продаје, набавке,...). Структура и понашање пословног система се моделирају преко **модела података** и **модела процеса** ИС-а. Развој ИС-а подразумева прављење модела процеса и модела података пословног система, који требају да буду реализовани у неком технолошком окружењу (оперативни систем, систем за управљање базом података, програмски језик,...). Модел процеса се описује помоћу Структурне Систем Анализе (ССА) [16]. Структурна систем анализа је представљена преко дијаграма токова података (ДТП). На основу дијаграма тока података могуће је направити речник података и дати прецизну спецификацију основних (примитивних) процеса система[17].

Прва фаза у развоју софтверског система, прикупљање захтева, се описује преко модела случаја коришћења (СК) који се добија на основу спецификације основних процеса. У фази анализе софтверског система, на основу модела СК се одређује модел података (структуре) и модел понашања софтверског система. Модел података софтверског система се описује помоћу проширеног модела објекти везе (ПМОВ), **релационог модела** (РМ), објектног модела (ОМ),..., итд. Релациони модел се може добити на основу ПМОВ-а. Модел понашања софтверског система се описује помоћу системских операција. Модел података и модел понашања софтверског система описују пословну логику софтверског система[18]. У фази пројектовања софтверског система[19] се дефинише тронивојска архитектура која се састоји од: корисничког интерфејса, апликационе логике и складишта података. Пројектовање сценарија коришћења екранских форми корисничког интерфејса се ради на основу СК-а, прецизније речено пројектовањем екранских форми се реализују случајеви коришћења. Кориснички интерфејс и апликациона логика се могу имплементирати коришћењем разних технологија (нпр.Java, C#, ...). Апликациона логика се прави на основу пословне

[15] Овде смо дали концептуални оквир развоја информационих и софтверских система који се изучава и примењује на ФОН-у.

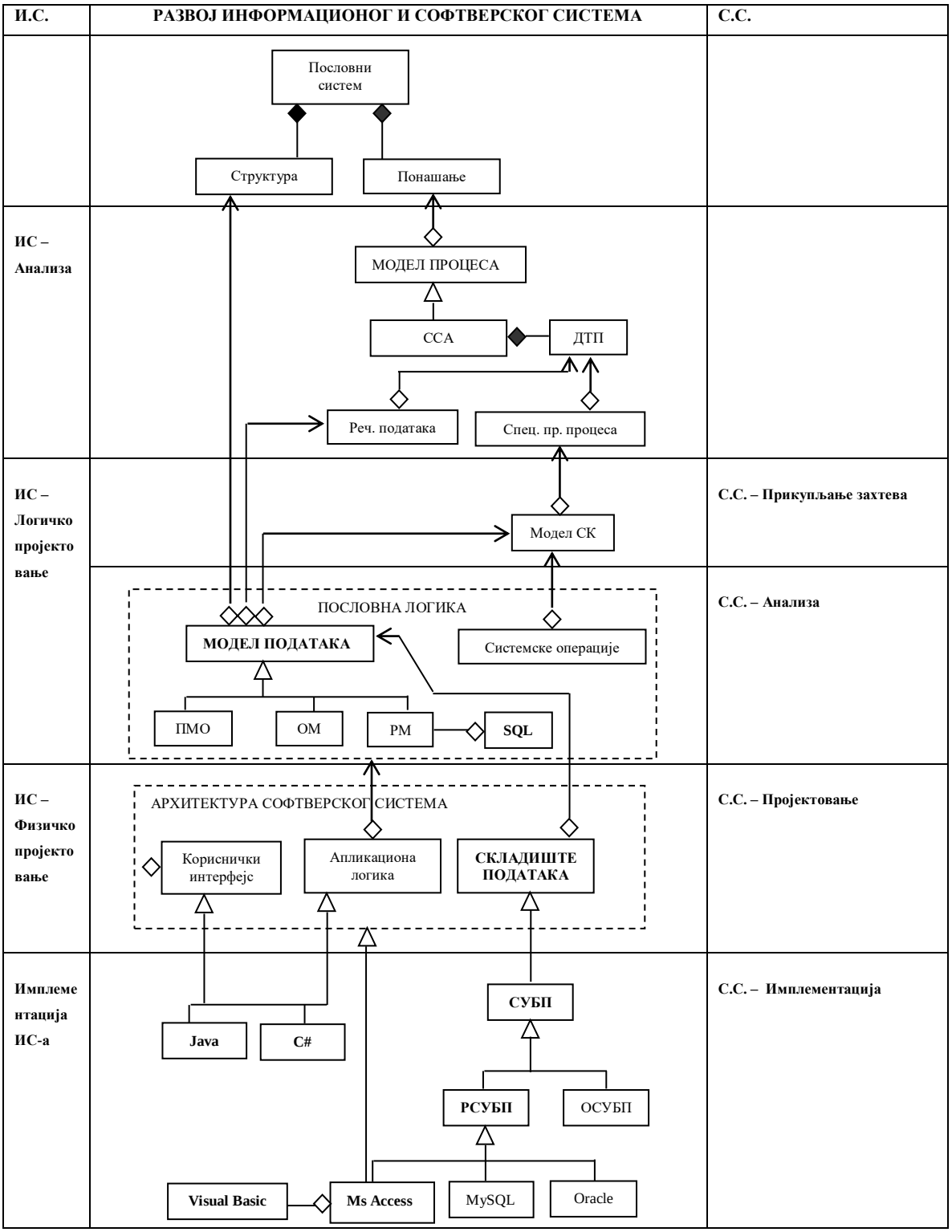
[16] Ово су неки од постојећих модела података и модела процеса.

[17] Структурна систем анализа се ради у **фази анализе ИС-а.**

[18] Фаза прикупљања захтева и анализа софтверског система се из перспективе ИС-а називају фаза **логичког пројектовања ИС-а**.

[19] Фаза пројектовања софтверског система се из перспективе ИС-а назива фаза **физичког пројектовања ИС-а**. Фаза имплементације софтверског система се из перспективе ИС-а назива фаза имплементације ИС-а.

логике софтверског система. У складишту података се чувају подаци. Складиште података може бити реализовано преко система за управљање базом података, системом датотека,...,итд. Системи за управљање базом података (СУБП) могу бити: релациони (РСУБП), објектни (ОСУБП),...,итд. Релациони СУБП су: **MS Access**, MySQL, Oracle,.., итд. Код MS Access-а се користи програмски језик **Visual Basic** као и **SQL** упитни језик помоћу кога правимо упите над табелама базе података. SQL упитни језик је заснован на релационом моделу. Помоћу MS Access-а је могуће направити веома богат кориснички интерфејс.



3.2 SQL упитни језик

SQL1: Основни појмови релационог модела

SQL (*Structured Query Language*)**),** структуирани упитни језик је заснован на релационом моделу. Релациони модел је дефинисан **структуром**, **операцијама** које се могу извршити над том структуром и **ограничењима** која морају да буду задовољена када се извршавају те операције[20].

Када будемо објашљавали SQL ми ћемо објаснити операције за дефиницију података (data definition), којима се креира: а) **база података,** б) **табеле** базе података и ц) **атрибути** табела базе података. Након тога ћемо објаснити операције за манипулацију подацима (data manipulation), које се могу поделити на операције: а) ажурирања: убаци (insert), промени (update), обриши (delete) и б) извештавања (select) над базом података.

Пре него што изложимо SQL, објаснићемо неке од кључних појмова релационог модела који су потребни како би се схватио SQL.

Релација дефинисана на n скупова је подскуп Декартовог производа тих n скупова. Појмови релација и табела биће посматрани као синоними како би се поједноставило схватање релационог модела. Пошто је релација скуп, а свака табела није скуп, дефинишу се услови које табела мора да задовољи да би била релација: а) Не постоје дупликати врста табеле, б) редослед врста није значајан и ц) редослед колона није значајан. Да би се могао дефинисати једноставан скуп операција над релацијама дефинише се допунски услов: д) Све вредности атрибута у релацијама су атомске. Нису дозвољени атрибути или групе атрибута са понављањем.

Уколико су задовољени наведени услови каже се да је релација у **Првој нормалној форми.** Свака релација у релационом моделу мора бити у првој нормалној форми.

Релација се састоји из атрибута. **Атрибут релације** се дефинише над неким доменом.

Домен релације у суштини представљају могуће типове атрибута релације.

Кардиналносат релације је број n-торки у релацији.

Примарни кључ јединствено идентификује једну n-торку у релацији или једну врсту у табели. **Прост примарни кључ** садржи један атрибут. **Сложен примарни кључ** садржи више атрибута

Особине примарног кључа:

а) Особине јединствености – Не постоје било које 2 n-торке са истом вредношћу примарног кључа.

б) Особина нередудантности – Ако се изостави било који део кључа губи се особина јединствености.

Може постојати више кандидата у релацији за примарни кључ. Онај кандидат који се изабере постаје примарни кључ док остали кандидати постају **алтернативни кључеви**.

Спољни кључ је атрибут или група атрибута у релацији који повезују ту релацију са другом релацијом.

Нула вредност се користи да означи још непознату вредност за неки од атрибута у неким n-торкама релације.

[20] У овој књизи нећемо се детаљније бавити структуром, ограничењима (вредносним и структурним) и операцијама над релационим моделом. Књига коју препоручујемо: **Базе података** [2]. У наведеној књизи је детаљно описан релациони модел.

SQL2: Дефиниција података (Data definition)

Пример CRDB1: Направити базу података по имену **racun**[21]**.**

```
CREATE DATABASE racun
```

Пример CRTAB1: Направити табелу: *racun* са атрибутима *BrojRacuna, NazivPartnera, UkupnaVrednost, Obradjen* и *Storniran*. Примарни кључ је атрибут BrojRacuna.

```
CREATE TABLE racun (
  BrojRacuna CHAR(10) PRIMARY KEY,
  NazivPartnera  CHAR(50),
  UkupnaVrednost  DOUBLE ,
  Obradjen YESNO,
  Storaniran YESNO
)
```

Пример CRTAB2: Направити табелу: *stavkaracuna* са атрибутима *BrojRacuna, RB, SifraProizvoda, Kolicina, ProdajnaCena* и *ProdajnaVrednost*. Примарни кључ је *BrojRacuna* и *RB*.

```
CREATE TABLE stavkaracuna (
  BrojRacuna CHAR(10),
  RB INTEGER,
  SifraProizvoda CHAR(20),
  Kolicina INTEGER,
  ProdajnaCena DOUBLE,
  ProdajnaVrednost DOUBLE,
  PRIMARY KEY  (BrojRacuna,RB)
  )
```

Пример ALTAB1: Додајте у табелу *stavkaracuna* атрибут *JedinicaMere* који је знаковног типа дужине 4.

```
ALTER TABLE stavkaracuna1
ADD COLUMN  JedinicaMere CHAR(4)
```

Пример ALDP2: Обрисати у табели *stavkaracuna* атрибут *SifraProizvoda*.

```
ALTER TABLE stavkaracuna
DROP COLUMN  JedinicaMere
```

Пример DRTAB1: Креирати табелу *test* са атрибутом *testatribut* и након тога је обрисати.

```
CREATE TABLE test(
  testatribut CHAR(10) PRIMARY KEY,
)

DROP TABLE test
```

[21] У Ms Access-у се прво ручно креира база података. Након тога се креирају табеле у бази података.

SQL3: Манипулација подацима (data manipulation)

Пример Insert1: Унети три слога рачуна и по неколико ставки рачуна за сваки рачун.

```
INSERT INTO Racun
VALUES ('1', 'Meridian invest D.O.O', 0, False, False);
-------------------------------------------------------------
INSERT INTO Racun
VALUES ('2', 'Perihard inženjering', 0, True, False);
-------------------------------------------------------------
INSERT INTO Racun
VALUES ('3', 'Sajam', 0, False, False);
 -------------------------------------------------------------
INSERT INTO StavkaRacuna
VALUES ('1', '1','pr1',5,45.00,0);
-------------------------------------------------------------
INSERT INTO StavkaRacuna
VALUES ('1', '2','pr2',1,45.00,0);
-------------------------------------------------------------
INSERT INTO StavkaRacuna
VALUES ('1', '3','pr3',2,56.34,0);
-------------------------------------------------------------
INSERT INTO StavkaRacuna
VALUES ('2', '1','pr1',7,12.56,0);
-------------------------------------------------------------
INSERT INTO StavkaRacuna
VALUES ('2', '2','pr2',7,12.00,0);
-------------------------------------------------------------
INSERT INTO StavkaRacuna
VALUES ('3', '1','pr1',3,45.00,0);
-------------------------------------------------------------
```

Пример Update1: Променити рачуне код којих је назив партнера "*Sajam*". Уместо тога ставити "*Beogradski sajam*"

```
UPDATE Racun SET NazivPartnera = "Beogradski sajam"
WHERE NazivPartnera="Sajam";
```

Пример Update2: Израчунати и запамтити продајне вредности ставки свих рачуна по формули: *ProdajnaVrednost = ProdajnaCena * Kolicina*

```
UPDATE StavkaRacuna SET
ProdajnaVrednost = ProdajnaCena*Kolicina
```

Пример Del1: Обрисати рачун број 3.

```
DELETE
FROM Racun
WHERE BrojRacuna= "3";
```

Пример Sel1: Приказати све рачуне (*приказати све слогове са свим атрибутима навођењем имена атрибута*)

```
SELECT BrojRacuna,NazivPartnera, UkupnaVrednost,Obradjen, Storniran FROM racun;
```

BrojRacuna	NazivPartnera	UkupnaVrednost	Obradjen	Storniran
1	Meridian invest D.O.O	0	N	N
2	Perihard inzenjering	0	N	N

Пример Sel2: Приказати све рачуне (*приказати све слогове са свим атрибутима преко* *).

```
SELECT * FROM racun;
```

BrojRacuna	NazivPartnera	UkupnaVrednost	Obradjen	Storniran
1	Meridian invest D.O.O	0	N	N
2	Perihard inzenjering	0	N	N

Пример Sel3: Приказати бројеве рачуна и називе партнера за све рачуне (*приказати све слогове за један или неколико атрибута навођењем имена атрибута*)

```
SELECT  BrojRacuna, NazivPartnera FROM racun;
```

BrojRacuna	NazivPartnera
1	Meridian invest D.O.O
2	Perihard inzenjering

Пример Dist1: Приказати различите бројеве рачуна са ставке рачуна (*приказати различите слогове табеле*).

```
SELECT DISTINCT BrojRacuna FROM StavkaRacuna;
```

BrojRacuna
1
2

Пример Dist2: Приказати различите редне бројеве рачуна (*Приказати различите слогове табеле*).

```
SELECT DISTINCT rb FROM StavkaRacuna;
```

RB
1
2
3

Пример Sel4: Приказати податке о рачуну чији је број 2 (*Приказати слогове табеле под неким условом*).

```
SELECT * FROM racun WHERE BrojRacuna = "2";
```

BrojRacuna	NazivPartnera	UkupnaVrednost	Obradjen	Storniran
2	Perihard inzenjering	0	N	N

Пример OrdBy1: Приказати редне бројеве ставки рачуна, у опадајућем редоследу, за рачун чији је број 1 (*Приказ слогова табеле у сортираном редоследу*).

```
SELECT * FROM StavkaRacuna WHERE BrojRacuna="1"
ORDER BY RB DESC;
```

BrojRacuna	RB	SifraProizvoda	Kolicina	ProdajnaCena	ProdajnaVrednost
1	3	pr3	2	56.34	112.68
1	2	pr2	1	45	45
1	1	pr1	5	45	225

Пример Max1: Наћи највећу количину код ставки рачуна чији је број 1 (*Функција за израчунавање максималне вредности из неког скупа вредности*).

```
SELECT Max(Kolicina) AS MaxKolicina
FROM StavkaRacuna
WHERE BrojRacuna="1";
```

MaxKolicina
5

Пример Sum1: Приказати укупну вредност рачуна чији је број 1 (*Функције за израчунавање суме неких вредности*).

```
SELECT Sum(ProdajnaVrednost) AS UkupnaVrednost
FROM StavkaRacuna
WHERE BrojRacuna="1";
```

UkupnaVrednost
382.68

Пример Count1: Наћи број ставки рачуна чији је број 1(*Функција за рачунање броја слогова у табели*).

```
SELECT Count(*) AS BrojStavki
FROM StavkaRacuna
WHERE BrojRacuna="1";
```

BrojStavki
3

Пример Count2: Наћи број ставки рачуна чији је број 1 по атрибуту ProdajnaCena (*Функција за рачунање броја слогова у табели*).

```
SELECT Count(ProdajnaCena) AS BrojStavki
FROM StavkaRacuna
WHERE BrojRacuna="1";
```

BrojStavki
3

Пример GroupBy1: Приказати колико је продато комада за сваку групу (врсту) производа (*Груписање слогова табеле по једном или више атрибута*).

За наведени пример упит се изводи над табелом ***StavkaRacuna***.

BrojRacuna	RB	SifraProizvoda	Kolicina	ProdajnaCena	ProdajnaVrednost
1	1	pr1	5	45	225
1	2	pr2	1	45	45
1	3	pr3	2	56.34	112.68
2	1	pr1	7	12.56	87.92
2	2	pr2	7	12	84
3	1	pr1	3	45	135

```
SELECT SifraProizvoda,Sum(Kolicina) as UkupnaKolicina FROM stavkaracuna Group By SifraProizvoda
```

SifraProizvoda	UkupnaKolicina
pr1	15
pr2	8
pr3	2

Пример GroupBy2: За сваки рачун показати колико је продато производа (сумарно) (*Груписање слогова табеле по једном или више атрибута*).

```
SELECT BrojRacuna,Sum(Kolicina) as UkupnaKolicina FROM stavkaracuna Group By BrojRacuna
```

BrojRacuna	UkupnaKolicina
1	8
2	14
3	3

Пример GroupBy3: За сваки рачун одредити број ставки (*Груписање слогова табеле по једном или више атрибута*).

```
SELECT BrojRacuna,Count(*) as BrojStavki FROM stavkaracuna Group By BrojRacuna
```

BrojRacuna	BrojStavki
1	3
2	2
3	1

Пример GroupBy4: За сваки производ одредити рачуне. Рачуне сортирати по шифри производа у растућем редоследу и количини у опадајућем редоследу (*Груписање слогова табеле по једном или више атрибута*).

```
SELECT SifraProizvoda, BrojRacuna, Kolicina
FROM stavkaracuna
GROUP BY SifraProizvoda, BrojRacuna, Kolicina
ORDER BY SifraProizvoda, Kolicina Desc
```

SifraProizvoda	BrojRacuna	Kolicina
pr1	2	7
pr1	1	5
pr1	3	3
pr2	2	7
pr2	1	1
pr3	1	2

Пример Having1: Приказати рачуне који имају број ставки већи од 1 (*Селекција слогова на основу услова о неком броју слогова*)

SELECT BrojRacuna,Count(*) as BrojStavki FROM stavkaracuna Group By BrojRacuna HAVING Count(*) > 1

BrojRacuna	BrojStavki
1	3
2	2

Пример ArFun1: Заокружити продајну вредност ставки рачуна на 1 децималу (*Аритметичке функције*)

SELECT RB,SifraProizvoda,Kolicina,ProdajnaCena, Round(ProdajnaVrednost,1) as ProdajnaVrednost1 FROM stavkaracuna

RB	SifraProizvoda	Kolicina	ProdajnaCena	ProdajnaVrednost
1	pr1	5	45	225.0
2	pr2	1	45	45.0
3	pr3	2	56.34	112.7
1	pr1	7	12.56	87.9
2	pr2	7	12	84.0
1	pr1	3	45	135.0

Пример ArFun2: За сваки рачун приказати број знакова назива партнера (*Функција за рад са стринговима*).

Select NazivPartnera, Len(NazivPartnera) as DuzinaNaziva From Racun

NazivPartnera	DuzinaNaziva
Meridian invest D.O.O	21
Perihard inzenjering	20
Sajam	5

Пример Ug1: Приказати рачуне код којих је продато више од 7 производа (*Угњеждени упити*) .

Пре него што се прикаже решење овога примера погледаћемо колико је по сваком рачуну продато производа:

Select BrojRacuna,Sum(Kolicina) as UkupnaKolicina FROM stavkaracuna Group By BrojRacuna

BrojRacuna	UkupnaKolicina
1	8
2	14
3	7

*Решење примера **Ug1**:*

Select BrojRacuna, suma From (Select BrojRacuna, Sum(Kolicina) as suma From stavkaracuna Group By BrojRacuna) as sume[22] Where suma>7

BrojRacuna	suma
1	8
2	14

[22] *sume* је назив унутрашњег упита. У суштини дешава се упит.

```
Select * From stavkaracuna Where sume Where suma>7
```

Пример Ug2: Приказати ставке рачуна које су направљене за *Perihard inzenjering* (*Угњеждени упити*).

```
Select * From stavkaracuna Where BrojRacuna = (Select BrojRacuna from Racun
Where NazivPartnera = "Perihard inženjering")
```

BrojRacuna	RB	SifraProizvoda	Kolicina	ProdajnaCena	ProdajnaVrednost
2	1	Pr1	7	12.56	87.92
2	2	Pr2	7	12	84

Правило: *Унутрашњи упит мора да врати само једну вредност ако се у Where клаузули користе оператори поређења (>=,<=,=,<,>)* **.**

Уколико би направили још један рачун за *Perihard inzenjering*:

```
INSERT INTO Racun VALUES ('4', 'Perihard inženjering', 0, False, False);
```

са једном ставком:

```
INSERT INTO StavkaRacuna VALUES ('4', '1','pr1',5,71.68,0);
```

тада би се при извршењу упита:

```
Select * From stavkaracuna Where BrojRacuna = (Select BrojRacuna from Racun
Where NazivPartnera = "Perihard inženjering")
```

јавила порука: *Subquery returns more than 1 row* (Подупит враћа више од 1 реда)

Тада се користи **IN** наредба уместо оператора = .

```
Select * From stavkaracuna Where BrojRacuna IN (Select BrojRacuna from Racun
Where NazivPartnera = "Perihard inzenjering")
```

BrojRacuna	RB	SifraProizvoda	Kolicina	ProdajnaCena	ProdajnaVrednost
2	1	pr1	7	12.56	87.92
2	2	pr2	7	12	84
4	1	pr1	5	71.68	0

Пример Ug3: Приказати рачун код кога је продато највише производа (*Угњеждени упити*).

```
Select BrojRacuna,suma From (Select BrojRacuna, Sum(Kolicina) as suma From
stavkaracuna Group By BrojRacuna) as sume Where suma = (Select Max(suma)
From (Select BrojRacuna,sum(Kolicina) as suma From stavkaracuna Group By
BrojRacuna) as sume)
```

BrojRacuna	suma
2	14

Друго решење:
а) Прво се направи упит *Sume*:

```
Select BrojRacuna, Sum(Kolicina) as suma From stavkaracuna GroupBy
BrojRacuna
```

б) Над упитом *Sume* се направи упит:

```
Select BrojRacuna,suma From Sume Where suma = (Select Max(suma) From
Sume)
```

Пример Uni1: Приказати све ставке рачуна 1 и 2 преко уније (*Унија две табеле*)**.**

```
SELECT * FROM stavkaracuna WHERE BrojRacuna = '1'
UNION
SELECT * FROM stavkaracuna WHERE BrojRacuna = '2'
```

BrojRacuna	RB	SifraProizvoda	Kolicina	ProdajnaCena	ProdajnaVrednost
1	1	pr1	5	45	225
1	2	pr2	1	45	45
1	3	pr3	2	56.34	112.68
2	1	pr1	7	12.56	87.92
2	2	pr2	7	12	84

Пример Raz1: Приказати рачуне свих пословних партнера осим *Perihard inzenjeringa*.

```
SELECT * FROM racun WHERE NazivPartnera <>"Perihard inzenjering"
```

BrojRacuna	NazivPartnera	UkupnaVrednost	Obradjen	Storniran
1	Meridian invest D.O.O	0	N	N
3	Beogradski sajam	0	N	N

Пример Equijoin1: Приказати за сваку ставку рачуна име партнера за који је везана ставка (*Спајање две или више табела - Спајање на једнакост (equijoin)*)

```
SELECT nazivpartnera,stavkaracuna.*  FROM racun,stavkaracuna WHERE
racun.BrojRacuna = stavkaracuna.BrojRacuna
```

Nazivpartnera	BrojRacuna	RB	SifraProizvoda	Kolicina	Prodajna Cena	Prodajna Vrednost
Meridian invest D.O.O	1	1	pr1	5	45	225
Meridian invest D.O.O	1	2	pr2	1	45	45
Meridian invest D.O.O	1	3	pr3	2	56.34	112.68
Perihard inzenjering	2	1	pr1	7	12.56	87.92
Perihard inzenjering	2	2	pr2	7	12	84
Beogradski sajam	3	1	pr1	3	45	135
Beogradski sajam	3	2		4	67.45	0
Perihard inženjering	4	1	pr1	5	71.68	0

Пример CarJoin1: Приказати све комбинације слогова табела *Racun* и *StavkaRacuna* (из предходног примера смо изоставили Where клаузулу) (*Спајање две или више табела - **Декартов производ (cartesian join)***)

```
SELECT nazivpartnera,stavkaracuna.BrojRacuna,stavkaRacuna.RB  FROM racun, stavkaracuna
```

nazivpartnera	BrojRacuna	RB
Meridian invest D.O.O	1	1
Perihard inzenjering	1	1
Beogradski sajam	1	1
Perihard inženjering	1	1
Meridian invest D.O.O	1	2
Perihard inzenjering	1	2
Beogradski sajam	1	2
Perihard inženjering	1	2
Meridian invest D.O.O	1	3
Perihard inzenjering	1	3
Beogradski sajam	1	3
Perihard inženjering	1	3
Meridian invest D.O.O	2	1
Perihard inzenjering	2	1
Beogradski sajam	2	1
Perihard inženjering	2	1
Meridian invest D.O.O	2	2
Perihard inzenjering	2	2
Beogradski sajam	2	2
Perihard inženjering	2	2
Meridian invest D.O.O	3	1
Perihard inzenjering	3	1
Beogradski sajam	3	1
Perihard inženjering	3	1
Meridian invest D.O.O	3	2
Perihard inzenjering	3	2
Beogradski sajam	3	2
Perihard inženjering	3	2
Meridian invest D.O.O	4	1
Perihard inzenjering	4	1
Beogradski sajam	4	1
Perihard inženjering	4	1

За 4 рачуна и 8 ставки рачуна има укупно 32 комбинације (парова) рачун-ставка рачуна.

Пример LeftJoin1: Приказати све шифре производа и ставке рачуна које су повезане са тим производима (*Спајање две или више табела - Спољно спајање (outer join) – лево спајање (left join)*)

Креираћемо табелу производ:

```
CREATE TABLE `racun`.`proizvod` (
  `SifraProizvoda` VARCHAR(20) NOT NULL,
  PRIMARY KEY(`SifraProizvoda`)
)
```

Унећемо неколико производа.

```
INSERT INTO proizvod  VALUES ('pr1');
INSERT INTO proizvod  VALUES ('pr2');
INSERT INTO proizvod  VALUES ('pr3');
INSERT INTO proizvod  VALUES ('pr4');
INSERT INTO proizvod  VALUES ('pr5');
```

Након тога ће се извршити упит:

```
SELECT Proizvod.SifraProizvoda, StavkaRacuna.BrojRacuna, StavkaRacuna.RB,
StavkaRacuna.SifraProizvoda
FROM Proizvod LEFT JOIN StavkaRacuna ON Proizvod.SifraProizvoda =
StavkaRacuna.SifraProizvoda;
```

SifraProizvoda	BrojRacuna	RB	SifraProizvoda
pr1	1	1	pr1
pr1	2	1	pr1
pr1	3	1	pr1
pr1	4	1	pr1
pr2	1	2	pr2
pr2	2	2	pr2
pr3	1	3	pr3
pr4			
pr5			

Правило: Приказују се сви слогови са леве (*LEFT*) стране од *JOIN* (Proizvod) и само они слогови са десне од *JOIN* (*StavkaRacuna*) који задовољавају услов: *Proizvod.SifraProizvoda = StavkaRacuna.SifraProizvoda*

Пример RightJoin1: Приказати све ставке рачуна независно од тога да ли су везане за постојеће шифре производа (*Спајање две или више табела - Спољно спајање (outer join) – десно спајање (right join)*).

```
SELECT Proizvod.SifraProizvoda, StavkaRacuna.BrojRacuna, StavkaRacuna.RB
FROM Proizvod RIGHT JOIN StavkaRacuna ON Proizvod.SifraProizvoda =
StavkaRacuna.SifraProizvoda;
```

SifraProizvoda	BrojRacuna	RB
pr1	1	1
pr2	1	2
pr3	1	3
pr1	2	1
pr2	2	2
pr1	3	1
	3	2
pr1	4	1

Правило: Приказују се сви слогови са десне (*RIGHT*) стране од *JOIN* (*StavkaRacuna*) и само они слогови са леве стране од JOIN (*Racun*) који задовољавају услов: *Proizvod.SifraProizvoda = StavkaRacuna.SifraProizvoda*

Пример SelfJoin1: Приказати хијерархију производа почев од производа *pr1* који је на врху (*Спајање две или више табела - Спољно спајање (outer join) – Само спајање (self join)*)

Проширићемо табелу производ са допунским атрибутом *Nadredjeni*.

```
ALTER TABLE `racun`.`proizvod` MODIFY COLUMN `SifraProizvoda`
VARCHAR(20) NOT NULL, ADD COLUMN `Nadredjeni` VARCHAR(20) NOT NULL;
```

Направићемо следећу хијерархијску саставницу (1 подређени производ улази у 1 надређени производ; у један надређени производ може да уђе више подређених производа). Производ *pr1* се састоји из производа *pr2* и *pr3*. Производ *pr2* се састоји од производа *pr4* и *pr5*.

```
UPDATE proizvod SET Nadredjeni = 'pr1' Where SifraProizvoda = 'pr2';
UPDATE proizvod SET Nadredjeni = 'pr1' Where SifraProizvoda = 'pr3';
UPDATE proizvod SET Nadredjeni = 'pr2' Where SifraProizvoda = 'pr4';
UPDATE proizvod SET Nadredjeni = 'pr2' Where SifraProizvoda = 'pr5';
UPDATE proizvod SET Nadredjeni = 'vrh' Where SifraProizvoda = 'pr1';
```

На крају ћемо извршити упит:

```
SELECT p.SifraProizvoda,p.Nadredjeni,s.SifraProizvoda as Podredjeni FROM proizvod p, proizvod s Where p.SifraProizvoda = s.Nadredjeni
```

SifraProizvoda	Nadredjeni	Podredjeni
pr1	vrh	pr2
pr1	vrh	pr3
pr2	pr1	pr4
pr2	pr1	pr5

Објашњење - *Табеле p и s се спајају преко p.SifraProizvoda = s.Nadredjeni на следећи начин*:

Табела *p*

SifraProizvoda	Nadredjeni
pr1	**vrh**
pr2	**pr1**
pr3	pr1
pr4	pr2
pr5	pr2

Tabela *s*

SifraProizvoda (ovo je Podredjeni	Nadredjeni
pr1	vrh
pr2	pr1
pr3	pr1
pr4	pr2
pr5	pr2

3.3 Обрада скупова слогова (recordsets)

Скупови слогова (recordsets) представљају n-торке (слогове) из табела база података. Скупови слогова се налазе у оперативној меморији, док се n-торке табела базе података налазе на спољној меморији. Над табелом или табелама базе података се извршавају упити који као резултат дају неки скуп n-торки које се чувају у *скупу слогова*. Над *скупом слогова* се могу изводити операције ажурирања или извештавања. Уколико се над скупом слогова изводе операције ажурирања, њихов резултат се памти у табелама базе података.

Примери

Примере које ћемо објаснити изводе се над табелама *Racun* и *StavkaRacuna*:

Racun				
BrojRacuna	**NazivPartnera**	**UkupnaVrednost**	**Obradjen**	**Storniran**
1	Meridian	1038,515412	Yes	Yes
2	Meridian	110200	No	No
3	Perihard	3138,9	No	No

StavkaRacuna					
BrojRacuna	**RB**	**SifraProizvoda**	**Kolicina**	**ProdajnaCena**	**ProdajnaVrednost**
1	1	s1	8	696,746857488	5573,974859904
1	2	s3	6	755,909908644	4535,459451864
2	1	s3	41	2200	90200
2	2	s1	40	500	20000
3	1	s1	11	1617,78	17795,58
3	2	s3	9	1200	10800
3	3	s1	8	1617,78	12942,24

CC1 – *Прочитати и приказати из табеле Racun (која је креирана у задатку CRTAB1) слогове.*

Private Sub **Command0_Click**()	Почетак процедуре.
Dim db As Database	Декларисање базе података.
Dim rs As Recordset	Декларисање *скупа слогова*.
Dim upit As String	
Set db = CurrentDb	Повезивање са текуће отвореном MS Access базом података.
upit = "Select * From Racun "	Дефинисање упита.
Set rs = db.OpenRecordset(upit, dbOpenDynaset)	Извршење упита и памћење резултата упита у *скуп слогова (rs).*
If rs.RecordCount = 0 Then	Провера да ли постоје слогови наведеног упита.
MsgBox "Ne postoji ni jedan slog u tabeli racun!!!"	Уколико не постоје слогови упита, приказује се порука о томе.
Else	Приказ свих слогова из rs.
rs.MoveFirst	Померање на први слог у *скупу слогова.*
While Not rs.EOF	Док се не прочита последњи слог извршава се циклична алгоритамска структура.
MsgBox "Broj racuna: " & rs!BrojRacuna & " -- Naziv partnera: " & rs!NazivPartnera & "--Ukupna vrednost: " & rs!UkupnaVrednost	Приказ слогова (рачуна)
rs.MoveNext	Померање на следећи слог у *скупу слогова.*
Wend	
End If	
End Sub	Крај процедуре.

Чита се табела рачун:

Racun				
BrojRacuna	**NazivPartnera**	**UkupnaVrednost**	**Obradjen**	**Storniran**
1	Meridian	1038,515412	Yes	Yes
2	Meridian	110200	No	No
3	Perihard	3138,9	No	No

и приказују се преко MsgBox функције сви слогови по атрибутима: *BrojRacuna*, *NazivPartnera* и *UkupnaVrednost:*

Broj racuna: 1 -- Naziv partnera: Meridian -- Ukupna vrednost: 1038,515412
Broj racuna: 2 -- Naziv partnera: Meridian -- Ukupna vrednost: 110200
Broj racuna: 3 -- Naziv partnera: Perihard -- Ukupna vrednost: 3138,9

CC2 – *Прочитати и приказати из табеле Racun све рачуне који имају 2 ставке рачуна.*

Private Sub **Command1_Click**()	Почетак процедуре.
Dim db As Database	Декларисање базе података.
Dim rs As Recordset	Декларисање *скупа слогова*.
Dim upit As String	
Set db = CurrentDb	Повезивање са текуће отвореном MS Access базом података.
upit = "SELECT BrojRacuna,Count(*) as BrojStavki FROM stavkaracuna Group By BrojRacuna HAVING Count(*) = 2"	Дефинисање упита.
Set rs = db.OpenRecordset(upit, dbOpenDynaset)	Извршење упита и памћење резултата упита у *скуп слогова (rs).*
If rs.RecordCount = 0 Then	Провера да ли постоје слогови наведеног упита.
MsgBox "Ne postoji ni jedan racun koji ima 2 stavke racuna!!!"	Уколико не постоје слогови упита приказује се порука о томе.
Else	
rs.MoveFirst	Померање на први слог у *скупу слогова.*
While Not rs.EOF	Док се не прочита последњи слог извршава се циклична алгоритамска структура.
MsgBox "Broj racuna: " & rs!BrojRacuna & "-- Broj stavki: " & rs!BrojStavki	Приказ слогова
rs.MoveNext	Померање на следећи слог у *скупу слогова.*
Wend	
End If	
End Sub ----->	Крај процедуре.

Чита се табела *StavkaRacuna*:

StavkaRacuna					
BrojRacuna	**RB**	**SifraProizvoda**	**Kolicina**	**ProdajnaCena**	**ProdajnaVrednost**
1	1	s1	8	696,746857488	5573,974859904
1	2	s3	6	755,909908644	4535,459451864
2	1	s3	41	2200	90200
2	2	s1	40	500	20000
3	1	s1	11	1617,78	17795,58
3	2	s3	9	1200	10800
3	3	s1	8	1617,78	12942,24

и приказују се преко MsgBox функције бројеви рачуна који имају 2 ставке рачуна:
Broj racuna: 1 -- Broj stavki: 2
Broj racuna: 2 -- Broj stavki: 2

***CC3** – Унети нови рачун (број рачуна је 4)у табелу Racun.*

Private Sub **Command2_Click**()	Почетак процедуре.
Dim db As Database	Декларисање базе података.
Dim rs As Recordset	Декларисање *скупа слогова*.
Set db = CurrentDb	Повезивање са текуће отвореном MS Access базом података.
Set rs = db.OpenRecordset("Racun", dbOpenDynaset)	Отвори табелу *Racun*.
rs.AddNew	Додај нови рачун у табелу *Racun*.
rs!BrojRacuna = "4"	
rs!NazivPartnera = "Meridian"	
rs!UkupnaVrednost = 0	
rs!Obradjen = False	
rs!Storniran = False	
rs.Update	
End Sub	Крај процедуре.

Након извршења процедуре табела *Racun* ће имати следећи изглед:

Racun				
BrojRacuna	**NazivPartnera**	**UkupnaVrednost**	**Obradjen**	**Storniran**
1	Meridian	1038,515412	Yes	Yes
2	Meridian	110200	No	No
3	Perihard	3138,9	No	No
4	Meridian	0	No	No

***CC4** – Обриши рачун број 4 у табели Racun.*

Private Sub **Command3_Click**()	Почетак процедуре.
Dim db As Database	Декларисање базе података.
Dim rs As Recordset	Декларисање *скупа слогова*.
Dim upit As String	
Set db = CurrentDb	Повезивање са текуће отвореном MS Access базом података.
upit = "Select * From Racun1 Where BrojRacuna ='4'"	Упит у коме се бира рачун са бројем 4.
Set rs = db.OpenRecordset(upit, dbOpenDynaset)	Извршење упита и памћење резултата упита у *скуп слогова (rs)*.
If rs.RecordCount = 0 Then	Провера да ли постоје слогови наведеног упита. Уколико не постоје слогови упита,
MsgBox "Ne postoji slog koji treba da se obrise!!!"	Приказује се порука о томе.
Else	
	Брише се слог (слогови). Уколико би било више рачуна са бројем 4, сви би били обрисани.
rs.MoveFirst	Померање на први слог у *скупу слогова*.
While Not rs.EOF	Док се не прочита последњи слог извршава се циклична алгоритамска структура.
rs.Delete	Брисање слога (рачуна).
rs.MoveNext	Померање на следећи слог у *скупу слогова*.
Wend	
End If	
End Sub -----> крај процедуре.	

Након извршења процедуре табела *Racun* ће имати следећи изглед:

Racun				
BrojRacuna	**NazivPartnera**	**UkupnaVrednost**	**Obradjen**	**Storniran**
1	Meridian	1038,515412	Yes	Yes
2	Meridian	110200	No	No
3	Perihard	3138,9	No	No

CC5 – *Промени рачун број 2, тако што се мења назив партнера (уместо Meridian уноси се Perihard) у табели Racun.*

Private Sub **Command4_Click**()	Почетак процедуре.
Dim db As Database	Декларисање базе података.
Dim rs As Recordset	Декларисање *скупа слогова*.
Dim upit As String	
Set db = CurrentDb	Повезивање са текуће отвореном MS Access базом података.
upit = "Select * From Racun1 Where BrojRacuna ='2'"	Упит у коме се бира рачун са бројем 2.
Set rs = db.OpenRecordset(upit, dbOpenDynaset)	Извршење упита и памћење резултата упита у *скуп слогова (rs).*
If rs.RecordCount = 0 Then	Провера да ли постоје слогови наведеног упита.
MsgBox "Ne postoji slog koji treba da se promeni!!!"	
Else	Мењају се слогови. Уколико би било више рачуна са бројем 2, сви би били промењени.
rs.MoveFirst	Померање на први слог у *скупу слогова.*
While Not rs.EOF	Док се не прочита последњи слог извршава се циклична алгоритамска структура.
rs.Edit	Промени код рачуна са бројем 2 назив партнера.
rs!NazivPartnera = "Perihard"	
rs.Update	
rs.MoveNext	Померање на следећи слог у *скупу слогова.*
Wend	
End If	
End Sub	Крај процедуре

Након извршења процедуре табела *Racun* ће имати следећи изглед:

Racun				
BrojRacuna	**NazivPartnera**	**UkupnaVrednost**	**Obradjen**	**Storniran**
1	Meridian	1038,515412	Yes	Yes
2	Perihard	110200	No	No
3	Perihard	3138,9	No	No

Задатак

ЗСС1 – *Креирати произвољну табелу у бази података и преко скупа слогова (recordset-а) извршити операције убаци, избаци, промени и прикажи слогове.*

5. Литература

1. Patricia Cardoza, Teresa Hennig, Graham Seach, Armen Stein: *Access 2003 VBA Programmer's Reference*, ISBN: 978-0-7645-5903-7, 2004.

2. Бранислав Лазаревић, Зоран Марјановић, Ненад Аничић, Слађан Бабарогић : *Базе података,* Факултет организационих наука, Београд, 2003.