ТЕМА 1. ПРОБЛЕМЕН АНАЛИЗ

В стопанската практика ежедневно възникват проблеми, които задължително трябва да се решат с помощта на компютър. Обикновено тези задачи се изпълняват на отделни етапи. При всички случаи е необходимо да се отдели време за анализиране на задачата по стъпки, за да се определи адекватен път за решение и неговата реализация. Естествено, че са необходими задълбочени познания относно възможностите на наличните програмни продукти. Преди всичко трябва да се вземе решение дали задачата може да бъде решена със стандартен пакет (и точно кой или няколко) или трябва да се направят собствени програми за решение.

Независимо от вида на проблема, необходимо е да се съблюдава следната последователност:

При поставяне на всеки проблем е необходимо да се направи обстоен анализ, преди да се премине към решаването му. Рутинните задачи съдържат проблеми, които вече веднъж са били анализирани и могат да се решат с вече изпитани методи. Задачите, които не са рутинни, трябва задължително да бъдат анализирани.

Следващите три стъпки ще помогнат за анализа на проблема така, че той или ще бъде решен или поне ще се установи, че той не се решава с наличните средства.

1.1. ИДЕНТИФИЦИРАНЕ НА ЗАДАЧАТА

Задачата трябва да се формулира възможно най-прецизно, като се определят и подзадачите, за които може да се приложи готово решение. Ако това не се направи, съществува опасност решението да се търси по дълъг и мъчителен път и нищо чудно въобще да не се стигне до добър резултат.

В първата фаза на проблемния анализ редом с видимите задачи трябва да се определят и тези, които стоят на по-заден план.

Пример: Отделът по маркетинг желае да изпрати серийни писма до клиентите си. Първостепенната задача е: с текстообработваща програма, за да се създаде писмо и след това да се отпечата и адресира до клиентите. Второстепенната задача, която се появява е в каква форма е съхранена информацията за клиентите. Дали данните са във формат, пригоден за текстообработващата програма или преди това се налага те да бъдат конвертирани?

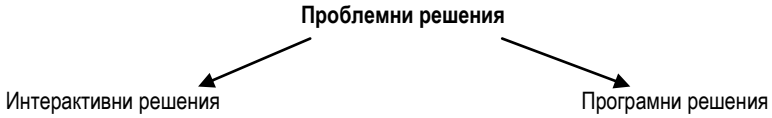
1.2. ИДЕНТИФИЦИРАНЕ НА ЕТАПИТЕ

Използваните техники за решаване на комплексни задачи обикновено се основават на пакет от задължителни условия. В този пакет детайлно се определят изискванията, на които програмата трябва да отговаря. Модерните методи описват изискванията към решението като сценарий от отделни етапи. Всеки сценарий описва стъпките, които потребителят трябва да направи, за да може компютърът да реши определена задача.

При описание на етапите трябва да се подхожда максимално детайлно. Колкото по-малки и елементарни са компонентите на сценария, толкова по-лесно ще се стигне до решението.

1.3. АНАЛИЗИРАНЕ НА ЕТАПИТЕ- Тук установените етапи трябва да бъдат детайлно описани стъпка по стъпка. Колкото повече са отделните детайли, толкова по-лесно ще бъде изпълнението на всяка стъпка.

1.4. АЛТЕРНАТИВИ ЗА РЕШЕНИЕ - За решаването на комплексни задачи с помощта на компютър съществуват основно две възможности:



Много от проблемите могат да се решават интерактивно, т.е. чрез стандартните програмни продукти без създаване на собствени програми. Понякога се налага да бъде използван стандартен софтуер за частично програмни решения. Макросите заемат междинна позиция. Те представляват списък от действия, които веднъж са определени, съхранени и при необходимост се изпълняват последователно. Просто казано, макросите са програми, чието създаване се поддържа от приложната програма и изпълнението им също е в рамките на това приложение.

Всички модерни софтуерни пакети имат средства и инструменти за създаване (автоматично), съхраняване и изпълнение на макроси.

Колкото повече се увеличават възможностите на хардуера, толкова повече са постиженията от софтуерна гледна точка. Стандартните пакети днес изпълняват задачи, които само преди няколко години е трябвало да бъдат програмирани самостоятелно.

За използването на модерните програми важи следното фаустовско правило: 90% от потребителите използват само 10% от предлаганите от програмата функции!

ТЕМА 2. ИНТЕРАКТИВНИ РЕШЕНИЯ С ПОМОЩТА НА MS ACCESS

Програмите за управление на бази от данни служат, от една страна, за съхраняване и обработка на големи масиви от данни, а от друга страна, подготвят входни данни за други програми и приложения. За пълноценна и смислена работа с MS Access е необходимо добре да се разбира релационния модел, а също и структурата на данните, които ще бъдат управлявани.

Основен закон на релационния модел е еднократното съхраняване на данните. За да се избягва повторното съхраняване на информация, данните се разполагат и управляват в няколко таблици, свързани помежду си с релации (отношения).

Access поддържа и система за защита на данните: при работа в мрежа с една база от данни се дефинират правата на всеки потребител. Така данните могат да бъдат защитени от неправоспособен потребител.

2.1. ПРОЕКТЕН ВИД НА ТАБЛИЦА И ЗАЩИТА НА ДАННИТЕ

При работа в интерактивен режим Access предоставя възможности за осигуряване и защита на данните още на етап проектиране на таблица.

Правило за валидност (Validation Rule): Ако това свойство на полето е установено в панела ***Field Properties,*** Access не приема данни, които не го изпълняват.

Пример:

Правило за валидност	Значение
>=0	Разрешава се въвеждане само на положителни стойности
>0 AND <1	Разрешава се въвеждане на стойности между 0 и 1
>#01.09.2004#	Датата трябва да бъде след 1-ви септември 2004 г.
="м" OR v	В полето се разрешава въвеждането само на стойности "м" или ж"
Between "1" AND "9"	В полето се въвеждат само цифрите от 1 до 9.

Съобщение за валидност (Validation Text). В това поле се задават съобщения за грешки, които Access трябва да изведе на екрана, когато е нарушено правилото за валидност.

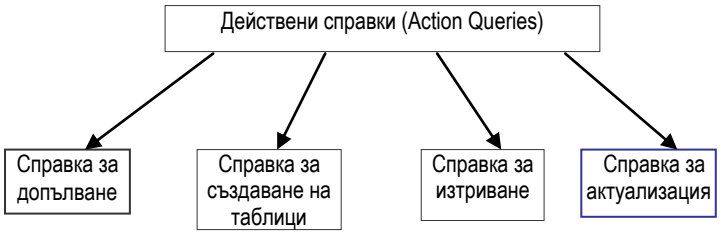
Задължително поле (Required). Ако това свойство е включено ***(Yes),*** Access изисква задължително въвеждане на данни в полето.

Стандартно значение (Default Value). Това свойство се използва за инициализиране на полета, които често имат една и съща стойност. Access автоматично задава в полето стандартната стойност,която разбира се, може да бъде поправена. Това свойство възпрепятства въвеждането на грешни данни и така осигурява защита на информацията.

2.2. ДЕЙСТВЕНИ СПРАВКИ (ACTION QUERIES)

Действените справки се отличават от селективните по това, че те извършват някакво действие, докато при селективните справки се показват само данни. По-точно казано, селективните справки показват като резултат други стойности, а не непременно тези, които се съхраняват в таблиците (проекция, селекция, изчисления). **Освен това, те не променят съдържанието на информацията.**

За разлика от тях, действените (активни) справки могат да променят данните в таблиците. Те са четири вида:



Справки за допълване *{Append Queries)*

С помощта на този вид справки могат да се добавят записи към вече готова таблица. Това е особено необходимо, когато за една база се въвеждат данни на различни компютри, които не са в мрежа. В един момент данните трябва да се обединят. Това може да стане чрез справка за допълване.

Справки за създаване на таблици *(Make Table Queries)*

Предназначението на тази справка е да създаде нова таблица от една или повече други таблици. Необходимостта за създаване на нова таблица се вижда от пръв поглед, когато в дадена таблица има излишно повторение на информация . Но не само в това е смисълът и целта на този вид справки. Те могат да изпълняват повече задачи:

» разделяне на *широка* таблица на две или повече обозрими таблици и установяване на релация 1:1 помежду им;

» прехвърляне на данни в друга таблица, когато тази не е проектирана *чисто;*

» изваждане на част от информацията, необходима на други хора за други задачи

Справка за изтриване *{Delete Queries)*

Със справките за изтриване могат да се изтриват както отделни записи, така и отделни полета. След проектиране на конкретната справка, чрез бутона ***Datasheet View*** от лентата за икони, може да се провери на кои записи действа справката за изтриване. Записите се изтриват едва след изпълнението на справката (кликване на бутона ***Run*** от лентата за икони или чрез бутона ***Open*** от основния екран на базата от данни).

ТЕМА 3. УПРАВЛЕНИЕ НА ДАННИТЕ С ПОМОЩТА НА ФОРМУЛЯРИ

Формулярите придават повече стил на екранното изображение на информацията. Те се използват за различни цели: най-често служат за по-пълно обхващане на данни от различни таблици. Чрез формулярите информацията от различни таблици може да се представи на екрана в прегледен вид. С тяхна помощ може да се направят потребителски менюта за отваряне или представяне на други формуляри и отчети. Освен това чрез тях става възможно създаването на диалогови полета и въвеждането на данни от потребителя за различни цели.

Данните на формуляра или отчета обикновено се съдържат в **контролни елементи.**

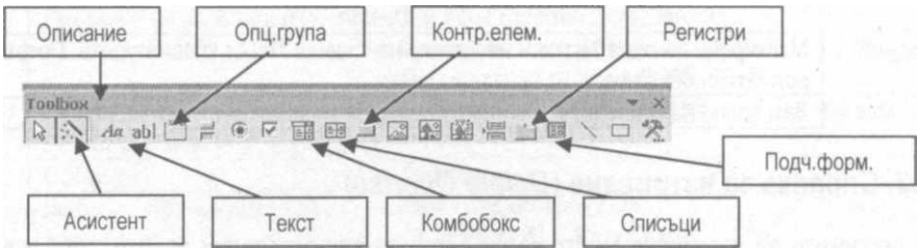
Access предоставя множество полезни контролни елементи,чрез които обработката на информацията значително се улеснява, намалява се възможността за допускане на неволни грешки, а работата с базата от данни става по-приятна и разбираема за обикновения потребител.

Контролните елементи могат да бъдат ***свързани, несвързани или изчислени.***

» Свързаният контролен елемент има установена връзка с определено поле от таблица или справка.

» В контролен елемент за изчисления може да се направи пресмятане на някаква стойност от базата (напр. 20% ДДС).

» Несвързаните контролни елементи нямат връзка с някое поле от таблица или справка и в тях може да се изобразяват картинки, чертежи или друга информация.



Обикновено за изработване на формуляр се използват асистентите за формуляри. След генериране на основните полета с помощта на контролните елементи от кутията за инструменти ***{Toolbox)*** във формуляра могат да се вградят различни елементи.

Списъчни и комбинирани полета - По-удобно и по-сигурно е дадена стойност да се избира от списък, вместо да се въвежда. При кликване върху дадена стойност в списък тя се въвежда автоматично в полето, свързано с този контролен елемент *{обвързано списъчно поле).*

Елементите на списъка, които се показват в поле, могат да се въведат или да се използва вече съществуваща таблица със стойности.

Ако списъчното поле не е обвързано, в него може да се съхранява стойност и да се използва при друг контролен елемент. В практиката необвързаните списъчни полета се използват при търсене на определени записи (напр. при търсене на участника г-н Филипов).

При въвеждане на данни чрез списъчно поле ***{Listbox),*** могат да се избират само стойности от списъка. Ако е необходимо да се даде възможност за въвеждане и на данни, неучастващи в списъка, трябва да се включи контролен елемент комбинирано поле ***{Combobox)*** - *комбинация от списъчно и текстово поле.* В този случай е възможно допълването на списъка с нова стойност.

Опционни групи *(Option Group)-* *с*ъздаването на опционни групи осигурява по-удобен избор от ограничен брой стойности. Опционните групи се състоят от рамка и група от опционни полета (радиобутони), контролни кутийки или командни бутони. При всички случаи се избира само **една** стойност. Ако трябва да се избира от по-голяма съвкупност от стойности, то по-добре ще бъде да се създаде списъчно или комбинирано поле.

Ако опционните полета, командните бутони или контролните кутийки се използват самостоятелно, това ще означава, че във формуляра ще се показват полета със стойност **Да/Не.** При значение **Да,** полето ще бъде маркирано, а при **Не,** ще бъде немаркирано.

Ако трябва да се избира от повече възможности, ще се наложи да се използва контролен елемент ***Option Group.*** Избраната от опционна група стойност се представя в свързаното поле от таблица като число. **Опционната стойност може да бъде само число, в никакъв случай текст!**

Регистрови контролни елементи *(Tab Control)*

Access предлага регистров контролен елемент за разпределяне на контролните елементи по регистри. По този начин може да се предотврати разполагане на контролните елементи от формуляра на няколко страници. За да се намери конкретна информация във формуляра няма да бъде необходимо да се превърта екранът, а просто ще се кликне на съответния регистър.**Важно!** Най-напред във формуляра трябва да се изтегли контролният елемент за регистри и едва тогава маркираните контролни елементи се разполагат (пласират) по регистри.

Контролен елемент - подчинен формуляр/отчет (Subform/Subreport)- подчиненият формуляр представлява формуляр, разположен в друг формуляр. Подчинените формуляри са много удобни за едновременно показване на данни от таблици или справки в отношение V.N. По този начин в подчинения формуляр (таблица в отношение N) могат да се покажат всички записи, които съответстват на запис от главния формуляр (таблицата в отношение1). Съответствието между главния и подчинения формуляр се осъществява чрез първичния и вторичния ключ.

ТЕМА 4. МАКРОСИ В ACCESS

Access е обектно-събитийно ориентирана програма. Всички обекти от екрана на Access могат да бъдат директно адресирани, а на всеки обект могат да бъдат присвоени свойства и събития с различни параметри. С макросите на Access се активират обектите ***Таблица, Формуляр, Справка, Отчет*** и става възможно създаване на професионално изработени приложения.

Създаване на макроси- *макросите в Access се създават при следната последователност:*

» Предварително се уточнява ситуацията, в която ще стартира макросът;

» Преди да се създаде макрос с определени действия е желателно тези действия да се проиграят и да се отбележат някои особености, ако има такива;

» Активира се обектът ***Macros*** от основния екран на базата от данни и се избират необходимите действия.

Макросите са част от базата от данни и затова при създаването им трябва да има отворена база от данни.

Изпълнение на макроси- след като се създаде макрос, неговото име е на разположение в обект ***Macros.*** Изпълнението на макросите не се различава от изпълнението на останалите обекти на Access,т.е.маркира се съответното име на макрос и се кликва бутон Run

Извикване на макрос от формуляр- С помощта на кутията с инструменти във формуляра се създават командни бутони. В тези случаи контролният елемент ***Асистент*** в ***Toolbox*** трябва да бъде *дезактивиран.* След кликване върху команден бутон, във формуляра се изтегля правоъгълник.*За изпълнение на макрос след кликване върху командния бутон трябва да се извършат следните стъпки:*

» Маркиране на контролния елемент във формуляра;

» View/Properties...

» В реда ***Caption*** се записва наименование за командния бутон;

» Превъртане на прозореца с характеристики до реда ***On Click*** и записване в него на името на макроса, който трябва да се изпълни. По-лесно може да си направи, ако се кликне в този ред, отдясно се появява стрелка и съответно списък с имената на съхранените макроси;

Когато пред една от буквите в наименованието на бутона (правилото е валидно само за латиница) се запише знакът

Action е стандартно действие, напр. отваряне или затваряне на формуляр. От предлаганите от Access действия трябва да се избере това, което отговаря на изискванията на задачата. Списък на всич-

&, то буквата след него се подчертава. Тогава командният бутон може да бъде активиран от клавиатурата с клавишна комбинация ***Alt+подчертаната буква.***

Създаване на собствено потребителско меню

За създаване на потребителско меню и собствена потребителска лента за меню трябва да се използва командата ***View/ Toolbars/Customize...Регистър: Toolbars/New...*** *Последователността на операциите е следната:*

- В диалоговото поле се задава име: **Изпълнение на макроси;**
- Към списъка от ивици за менюта е добавена нова празна ивица - ***Изпълнение на макроси;***
- Чрез кликване на бутон ***Properties*** се отваря диалоговият прозорец ***ToolbarsProperties,*** в който се определя дали новата ивица ще съдържа икони или ще бъде лента за меню, както и някои други свойства.
- Новата ивица се запълва с менюта, като се кликва на регистър ***Commands,*** категория ***АП Macros.***
- Имената на показаните, съхранени в базата макроси се изтеглят в новата лента за меню. (За тази цел лентата, която обикновено се крие зад диалоговия прозорец трябва да се пласира до него).
- За надписване или смяна на иконите на бутоните може да се ползва контекстното меню (показалецът на мишката е позициониран върху елемент от новата ивица, кликване с десен бутон). По стандарт Access поставя икона пред текста. Този стил може да се промени чрез контекстното меню. Да се избере стил на представяне ***само текст*** и да се промени големината на новата ивица.
- Ако новата ивица принадлежи на отчет или на формуляр, то тогава в свойството ***Menu Bar*** от ***Properties...*** на съответния обект трябва да се запише нейното име. Ако новата ивица или лента трябва да принадлежи на

всички обекти в базата от данни и да се намира на мястото на ивицата ***Standard,*** то тогава това се отбелязва в рубриката ***Menu Bar*** на диалоговото поле на командата ***Tools/ Startup...***

Използване на готови макроси- Access предлага голям брой вградени макроси, които могат да се използват във формуляр (напр. нов запис, последен запис и т.н.). Ако е необходимо използването на тези макроси, преди създаването на команден бутон трябва да се активира асистентът за контролни елементи (магическата пръчица) от ***Toolbox.*** След изчертаването на правоъгълник за команден бутон във формуляр се отваря следното диалогово поле, с което се работи по указанията на асистента:

Използване на макроси за търсене- Много често в една база от данни се налага да се търси конкретен запис или поле от таблица. С макросите на Access тази задача е лесно изпълнима. Макросът за търсене на поле от таблица се състои от:

1. Чрез ***GoToControl*** се определя полето за търсене;
2. Чрез ***Find Record*** се стартира търсенето.

ТЕМА 5. ИЗГРАЖДАНЕ НА VISUAL BASIC - ПРОГРАМА

С помощта на програмните пакети Excel, Access и Word решаването на различни проблеми от практиката протича по правило интерактивно. Всяка от тези програми съдържа и език за програмиране, чрез който могат да се решат проблеми, за които функционалните средства на продуктите не са достатъчни. Обикновено се търси комбинирано решаване на проблемите, като езикът за програмиране се ползва за по-специализирани решения.

VBA (Visual Basic for Applications) е език за програмиране на приложните програми на MS Office - Excel, Access, Word и PowerPoint. С помощта на това средство се разширяват функционалните възможности на стандартните пакети.

ВЪВЕДЕНИЕ- програмата представлява последователност от команди (оператори), които са разбираеми за компютъра (по-точно за компилатора), чрез които той изпълнява определени задачи.

При създаването на компютърни програми се ползват ***три елементарни структури****, а именно:*

» Последователна - отделните оператори се изпълняват последователно един след друг;

Пример: Въвеждане на единична цена
Изчисляване на евентуални разходи и загуби (надбавка)
Сумиране на единична цена и надбавка
Изчисляване на печалба
Формиране на продажна цена
Отпечатване на продажна цена

» Селективна *(чрез избор)* - операторите се изпълняват при изпълнение на дадено условие;

» Итерационна (циклична*)* - операторите се изпълняват отново;

ИЗГРАЖДАНЕ НА VISUAL BASIC-ПРОГРАМА- VBA принадлежи към *процедурните езици за програмиране,* като същевременно е и обектноори-ентиран. Характерното за тези езици е, че програмният код, т.е. операторите, се въвеждат в малки, делни части на програмата, наречени процедури. Оттук следва, че програмата на VBA се състои от една или повече процедури.

Процедурите (подпрограми) имат следния синтаксис:

```
Sub Име_на_процедура()
Оператори
End Sub
```

Процедурата винаги започва с ключовата дума `Sub` и завършва с `End Sub`. Между тези два оператора се намира тялото на процедурата, т.е. това са оператори, които се изпълняват след стартиране на процедурата.

Има три вида процедури:

а) *Синтаксис на Sub - процедури*

```
[Public | Private] SubИме_на_процедура()
Оператори
End Sub
```

`Sub` - процедурите се използват, когато трябва да се изпълнява последователност от оператори. Извикването им в повечето случаи не е свързано с настъпването на някакво събитие.

б) *Синтаксис на процедури - функции*

```
[Public | Private] Function Име_на_процедура (Аргументи) As Typ
 Оператори
End Function
```

Аргументите са стойности, които се предават на функцията за обработка. В частта `As Typ` се определя типът данни (напр. цяло или дробно число) за резултата от работата на функцията. Функцията винаги връща на извикващата програма резултат в променлива. Името на тази променлива съответства на името на процедурата.

Процедурите-функции са дефинирани от потребителя функции. Те могат да бъдат приравнени към стандартните функции от продуктите на MS Office, напр. `Sum () , Max () , Min () , Date ()` . Процедурите-функции се използват, когато трябва да се направи определено пресмятане, т.е. вместо отново да се изписва последователността от оператори, тя вече е въведена в дадена функция и при необходимост се извиква.

в) *Събитийни процедури*

Събитийните процедури (`Event`-процедури) съдържат оператори, които се изпълняват при настъпването на дадено събитие. Събитията обикновено са свързани с контролни елементи (командни бутони, текстови полета) от формуляр. Като събития се определят действия като кликване с мишката, натискане на клавиш от клавиатурата и др.

Синтаксис:

```
[Private] Sub Име-контролен-елемент_Име-събитие
 Оператори
End Sub
```

Пример:

```
Private Sub CommandButtonl_Click
MsgBoxf("Hallo")
End Sub
```

Обяснение: След кликване с мишката върху команден бутон CommandButtonl ще се изпълнят операторите от процедурата, в случая на екрана в кутия за съобщения ще се изпише `Hallo.` Ключовите думи **`Public`** и **`Private`** определят областта на приложение (валидност) на процедурите. Ако не е определен типът на процедурата, по подразбиране тя се приема като **`Public.`**

ТЕМА 6. ЕЛЕМЕНТИ НА ЕЗИКА ЗА ПРОГРАМИРАНЕ

В електронно-изчислителната техника се използват два основни типа данни: числови данни (числа) и буквено-цифрови данни (знаци, символни низове). Тези два типа данни могат да се използват в програмата като променливи и константи.

Променливи - Работната памет на компютъра се състои образно казано от еднакво големи номерирани клетки, които са лесно достъпни по техния номер (адрес). В една клетка може да бъде записан 1 байт информация. Следващата дефиниция изразява най-съществените характеристики на променливите:

***Променливите са клетки от паметта*,**

» обръщението към които се осъществява по име;

» чиято стойност (съдържание) може да бъде задавана или променяна по време на изпълнението на програмата;

» на които е присвоен определен тип данни (напр. Integer, Single, String). От този тип зависи какъв вид данни могат да се съхраняват в тях.

Имената на променливите са подчинени на следните правила:

» Имената могат да се състоят от букви, цифри и долна черта(_).Всички останали знаци са неприемливи;

» Първият символ на името трябва да бъде буква;

» Максималната дължина на името е 255 символа;

» Не се прави разлика между малки и главни букви. Имената ***NUM*** и ***Num*** са идентични;

» Служебните (резервирани) думи като напр. `Sub`,`MsgBox`, `End` не могат да бъдат имена на променливи;

» Желателно е да се избират говорящи и смислени имена.

Прим&р: Променлива за съхраняване на оборот не трябва да има име x, а напр. `Oborot` или `obrt`.

За какво се използват променливите? По време на изпълнение на програмите въведените данни, както и получените междинни резултати (суми, изчислени части като процент от цяло и др.) трябва да бъдат съхранени за по-нататъшна обработка или за отпечатване, било то на екран, принтер или файл. За тези цели се използват променливи.

Пример: От клавиатурата се въвеждат три числа. За да може те да бъдат използвани в програмата или отпечатани, трябва да се съхранят в променливи.

В следващия пример нагледно са представени променливи и числата, съхранени в тях:

		Num1 ***34***	***Num2*** ***56***		
				Num3 ***124***	

На променливите с имена `Num1, Num2, Num3` съответно са присвоени стойностите 34,56 и 124

ТЕМА 7.ТИПОВЕ ПРОМЕНЛИВИ

Програмистът на **Visual Basic** разполага със следните типове променливи:

Тип	Знак за този тип	Място в паметта	Описание	Минимална и максимална стойност
Byte	Няма	1 Byte	Цяло число	0-255
Integer	%	2 Byte	Цяло число	-32768 до+32767
Long	**&**	4 Byte	Цяло число	-2 147 483 648 до + 2 147 483 647
Single	!	4 Byte	Дробно число	-3.402823E+38 до -1,401298E-45 (отрицателни числа) +1.401298E-45 до 3.402823E+38 (положителни числа)
Double	#	8 Byte	Дробно число	-1,79769313486232E+308 до -4,94065645841247E-324 (отрицателни числа) +4,94065645841247E-324 до +1,79769313486232E+308 (положителни числа)
Currency	**@**	8 Byte	Дробно число	-922337203685477,5808 до +922337203685477,5808
String	**$**	1 Byte	Низ от знаци	
Date	Няма	8 Byte	Дата	01.01.100г. до 31.12.9999г.
Obj ect	Няма	4 Byte	Препратка към обект	
Variant	***Няма***		***Дата/време, число, символ***	***/8.45; 20.45***

На фона на различните типове прсг/екпиви трябва да се мисли и за въвеждане, съхраняване и обработка на какви данни става дума - цели числа, дробни числа, символни низове или от тип дата/време.

За съхраняване на цели числа **Visual Basic** предоставя типовете `Byte, integer` и `Long.` В компютъра те са представени като 8-, 16- и 32-битови двоични числа и заемат в паметта съответно 1, 2 или 4 байта.

При обработка на дробни числа в програма, трябва да се изберат типове променливи като `Single, Double ИЛИ Currency.` За числа `С` максимум 6 цифри е достатъчен `ТИП Single` (обикновена точност). За точната обработка на числа с повече позиции трябва да се ползва тип `Double` (двойна точност) или `Currency.` При използване на тип `Currency,` числата могат да имат максимум 15 знака в цялата част на числото и максимум 4 знака след десетичната точка. Бързата обработка и голямата точност правят този тип данни много удобен за работа с парични стойности.

ПРИМЕР:

Въведено число	**Съхраняване**	
	Обикновена точност	**Двойна точност**
Num! = 451,25	451,25	
Num! = 1,2345678	1,2345678	
Num! = 0,00123456789	1.2345678E-03	
Num# = 0,00123456789		0,00123456789
Num# = 12345678900512		12345678900512
Num#= 12345678900512351234		1,234567890051235E+19

Решението за използване на един или друг тип променлива зависи от големината на числото, което променливата трябва да приеме. За числа между 1 и 100 е достатъчен тип `Byte.` За дробни числа с 9 знака (напр. балансови суми) трябва да се избере тип `Double` с двойна точност.

Трябва да се обърне внимание и на различната скорост за обработка на различните типове. Най-бързо се обработват числата от тип `Byte`. Най-бавно се работи с изрази, съдържащи 8-байтови променливи.

Забележка: Обемът на заетата памет зависи от типа на променливите, а не от големината на съхранените числа. Съхраненото число 1 в променлива от тип `Long` заема 4 байта, въпреки че е достатъчен един байт и променлива от тип `Byte.`

Наред с числовите променливи могат да се използват и текстови или променливи от тип string, когато трябва да се въвеждат знаци или текст. Променливите от този тип могат да поемат до 65 000 знака (букви, цифри, специални символи). Всеки знак заема 1 байт от паметта и се съхранява в **ANSI**-код.

За данни от тип дата/време се избира тип променлива `Date.` Малко по-особена е променливата от тип `Variant.` Чрез определянето на тип за променливите се знае какви стойности се съхраняват в тях. Ако за дадена променлива не се зададе тип, то по подразбиране за нея се определя тип `variant.`

Характерното за променливите от тип `Variant,` че в тях могат да се съхраняват различни видове данни, като числа, символни низове, данни от тип дата/време. Променливите от този тип се определят едва след като в тях се запишат съответните данни. Необходимият обем памет за този тип променливи също няма фиксирана дължина и зависи от данните.

Ако в променлива от тип `variant` се съхраняват числа, **Visual Basic** употребява най-подходящия тип и възможния минимум памет. Това означава, че малки числа без дробна част ще се запишат в променлива от тип `Byte.` Ако на същата променлива по-късно в хода на програмата се присвои по-голяма стойност, **Visual Basic** ще смени типа на променливата от `Byte` на `integer` или `Long.` Ако зададената стойност е много голяма, или се появи и дробна част, тогава ще бъде избран `Double` - формат.

`Пример:`

Въведена/ Присвоена стойност	**Действие**
Стойност = 20	Променливата от тип `Variant` съдържа числовата стойност 20; съхранява се в `Byte` - формат.
Стойност = "U" & Стойност	Чрез знака & към стойността се присъединява (конка-тенира) буквата U. Новото съдържание е U20, т.е. това е дълъг символен низ.
Стойност = "ДДС"	В променливата се записва символният низ ДДС.
Стойност = 10520630,55	Символният низ е заменен от дробно число; то ще бъде съхранено в `Double`-формат.

Променливите от тип `variant` изглеждат практични за употреба. Програмистът не трябва сам да избира тип за променливите, типът се определя автоматично в зависимост от данните. Тази концепция има един много основен недостатък. При разчитане на програмния код програмистът никога не знае с точност от кой тип е дадена променлива

и каква стойност съдържа, което я прави непрегледна. При обработката на данните и извършването на операции с тях, използването на променливи от тип `variant` може да доведе до трудно откриваеми грешки.

От гледна точка на прегледността и отбягване на ненужни грешки в този учебник се дава предимство на предварителното точно и прецизно определяне типа на променливите.

ТЕМА 8. ДЕКЛАРИРАНЕ НА ПРОМЕНЛИВИ

При деклариране на променливите се задава тяхното име и тип. Напр. променливата се нарича `Ime` и има `ТИП String.`

След стартиране на готовата програма, операционната система резервира памет за променливите според декларирания за тях тип. С декларациите се определя и видът на данните, които се съдържат в тях. Напр. според декларирането променливата Ime може да съдържа символен низ, но не и число.

Чрез командата `Option Explicit` се определя и областта на действие (валидност) на използваните променливи в процедурата. Използването на тази команда означава, че **Visual Basic** ще изисква деклариране на всички използвани променливи и издава съобщения за грешка при недекларирани променливи. След *изчистването* на този пропуск изпълнението на програмата може да продължи.

Операторът **`Option Explicit`** помага за бързо откриване на недекларирани или грешно записани променливи и затова е препоръчително да се използва във всички програми. За автоматичното включване на оператора в областта за декларации на всички нови формуляри и модули е необходимо да се проведат следните действия:

» Превключване към **Visual Basic-Editor;**
» Активиране на ***Tools/Options...;***
» Избор на регистър ***Editor,***
» Активиране на контролната кутийка ***Require Variable Declaration;***
» Кликване на ***OK***

Забележка: Направената настройка не се отразява на другите налични програми.

» За деклариране се използва командата:
`Dim` Име_променлива **`[As тур]`**
» Ако не се зададе **`[As Тур],`** променливата се приема от тип **`Variant.`**

`Примери:`

Команда	**Действие**
Dim Ime **As String**	Променливите Ime, Vnoska, 1, Zena ще бъдат определени като **String-, Currency-, Integer-, Single-** променливи.
Dim Vnoska **As Currency**	
Dim 1 **As Integer**	
Dim Zena **As Single**	

В една команда могат да се направят повече декларации

Dim Num1,Num2 As **Integer**	Num1, Num2 са декларирани като променливи от тип **Integer.**
Dim Rez As **Double,** BroJAs **Integer**	Rez е декларирана като **Double** - променлива, а Broj като променлива от тип **integer.**

При деклариране на променливи от тип **`string`** трябва да се обърне внимание на следното. По подразбиране текстовите променливи имат различна дължина, т.е. дължината им и съответно резервираната за тях памет зависи от въведените или присвоени данни.

Примери:

DimFamiliaAs String

а) Familia = "Антова"
б) Familia = "Каранджулова"

В случай а) променливата заема 6 байта, а в случай б) 12 байта.

Друга алтернатива е декларирането на променливи с постоянна дължина. В този случай се използва следния синтаксис:

Команда: **Dim** Име_променлива As **String*Length**

Пример: Променливата Frame е декларирана с постоянна дължина от **30** знака (байта). Ако символният низ съдържа по-малко знаци, дължината му се допълва с интервали до **30,** ако съдържа повече от **30** знака те се отрязват и се съхраняват установените **30.**

Dim Frame As **String"30**

Служебните думи **As Typ** могат да се заменят със специалния знак за съответния тип:
Dim Ime$
Dim Vnoska®
Dim I%
Dim
Zena!

ТЕМА 9. ПРИСВОЯВАНЕ НА СТОЙНОСТИ

Много често след извършване на изчисления в дадена програма резултатът се присвоява на променлива по следния начин:

[Let] Променлива = Израз

Изразът може да бъде константа, променлива или комбинация от двете.

Аритметичният израз съдържа операнди ; това са числови константи или променливи, свързани чрез математическите оператори +; -; *; /; л. Чрез скоби се определя последователността за извършване на действията.

Операторът за присвояване [=] означава, че стойността отдясно на оператора ще се присвои на променливата отляво. Старата стойност на променливата се губи.

***Работа с** променливи:*

Пример1:

Ключовата дума **Let** не е задължителна и затова е оградена в квадратни скоби.

Присвояване	**Действие**
Num1 = 15 (1)	Променливата Num1 получава значение 15.
Num2 = 36 (2)	Променливата Num2 получава значение 36.
Num1 = Num2 (3)	На променливата Num1 се присвоява стойността на променливата Num2.
Sum = Num1 + Num2	На променливата Sum се присвоява резултатът от събиране на стойностите в Num1 и Num2.
Msg = "Имате грешка"	Променливата Msg ще съдържа константата "Имате грешка".
Rez = (Sum1 -Sum2)/12*3	На променливата Rez се присвоява резултатът от изчислението на аритметичния израз.
Suma = Suma + Num	Значението на променливата Num се прибавя към старото значение на Suma. Променливата Suma вече има ново значение.
Broj - Broj + 1	Стойността в променливата Broj се увеличава с 1.

Пример2:

1. Да се пресметне сумата на две числа и резултатът да се покаже на екрана в ***MsgBox.***

Решение:

```
Public Sub PromenliwM
  Dim Num1, Num2, Rez As Integer
  Num1 = 77 Num2 = 101 Rez = Num1 +
  Num2
  MsgBox ("Сумата на числата е:"+ Str(Rez))
  End Sub
```

Забележка: Функцията str () превръща числова стойност в символен низ!

ТЕМА 10. ПРОМЕНЛИВИ И ПРОЦЕДУРИ

Освен име, тип на данните и значение, променливата притежава и област на действие (валидност). За процедурите също трябва да бъде ясно дали могат да се извикват от определени модули, или са валидни за целия проект.

ЛОКАЛНИ ПРОМЕНЛИВИ- Локалните променливи се декларират в рамките на една процедура и могат да се използват само в нея. Използват се предимно за съхраняване на междинни резултати или за броячи с локално значение. След като се изпълнят командите на тази процедура, локалните променливи се изтриват от паметта. При ново повикване на процедурата, за тях отново се резервира памет.

ПРОМЕНЛИВИ И МОДУЛИ- Декларираните в частта за декларации на модула променливи могат да бъдат използвани само за този модул, но не и за останалите модули в рамките на проекта. Променливите на модула остават в паметта през цялото време от изпълнението на програмата.

ГЛОБАЛНИ ПРОМЕНЛИВИ- глобалните променливи се декларират в отрязъка за декларации на модула чрез използване на следния синтаксис:

`Public` Име_на_променлива `[As Тип]`

Този тип променливи имат област на действие в процедурите на всички модули на програмата. Те остават в паметта през цялото време от изпълнението на програмата и не губят стойността си.

ОБЛАСТ НА ДЕЙСТВИЕ НА ПРОЦЕДУРИ

Областта на действие на процедурите се дефинира чрез ключовите думи `Public` и `Private`.

`Public` - разширява областта на действие в рамките на целия проект. Това означава, че. процедурата може да се извиква от процедурите на същия модул или от процедури на други модули в рамките на един проект.

`Private` - ограничава действието на процедурата само до модула, в който е дефинирана. Извикването на процедури се осъществява чрез задаване на нейното име и евентуално пред него може да присъства и командата `Call`.

`[Call]` Име_на_процедура [(Списък от аргументи)]

Изписването на оператора `Call` не е задължително, но с него програмата става по-разбираема и удобна за четене. Извикване на процедура означава, че управлението на програмата се предава на нея. След изпълнение на командите в тази процедура, управлението се връща обратно в извикващата процедура и изпълнението продължава от оператора, който е кодиран след `Call`.

Константи

Константите са числа, знакови низове или данни от тип дата/време, които не се променят по време на изпълнението на програмата.

Пример: DDS = Zena * `0.2;` в случая `0.2` е константа и не се променя.

Администрирането и поддържането на програмата ще се улесни, ако се използва операторът `Const:`

`[Public]Const` Име_на_константа = `Стойност`

Име_на_константа е символично име, за което важат правилата за имена на променливи. Стойност може да бъде число, символен низ и др.

За деклариранеto и областта на действие на константите важат правилата за променливите. Има локални константи, глобални константи или такива, които са валидни само за един модул.

Примери:,

```
Const PI= 3.14159265
Public Const Maximum =6
Const Codename = "Ivan"
Const Naselenie_Zemja -7E+09
Const Zena_km = 0.39 '= Цена за километър
```

Коментари

Използването на коментари говори за добър стил на програмиране. Коментарите започват със знака апостроф (') или с командата **`Rem`.**

Забележка:

» Въведените с апостроф (') или с командата **`Rem`** коментари се оцветяват в зелено от **VBA;**

» При изпълнение на програмата, коментарите се игнорират

Пример: DDS = Zena*0.2 'ДДС е 20% от цената

ТЕМА 11. ИЗЧИСЛЯВАНЕ В VBA

В езика за програмиране **VBA** се използват познатите аритметични оператори +; -; *; /;^. При работа с аритметични изрази с повече операции са в сила приоритетите на аритметичните действия, познати от математиката.

Пример:

Оператор	**Резултат**
Debug. Print 6+2*5	16
Debug. Print (6+2)*5	40
Rez = 2000 + (Nom*1.20)	2600 (Norn = 500)
Rez = 7+3*4/6-(8+2)/5	7

Забележка: **Debug. Print** е основен оператор за извеждане на данни; използва се предимно при тестване на програма. Извеждането на данните е в директния прозорец, който се активира чрез меню ***View*** във **Visual Basic-Editor.**

Синтаксис:Debug.Print Израз

Математически функции и функции за конвертиране

Наред със самостоятелно програмираните функции има много готови, интегрирани във **VBA** функции. Касае се за така наречените **DLL**-файлове (Dynamics Link Libraries, т.е. библиотеки), в които се съдържат функции и процедури за определени области и са винаги на разположение при необходимост. Става дума за малки програми, които функционират на следния принцип:

Функцията се извиква по име, като при това в скобите се предава поне един аргумент. Аргументът (по правило числова стойност) се обработва от функцията. Като резултат на извикващата процедура се предава само една стойност (числова).

МАТЕМАТИЧЕСКИ ФУНКЦИИ

Sqr изчислява квадратен корен от число

Синтаксис: Променлива = **Sqr**(Число)

Debug. Print Sqr(Число)

При извикване на функцията **Sqr** се предава аргумента Число. Функцията пресмята квадратния корен и предава резултата на променлива или директно се показва в директния прозорец.

Rez = **Sqr (25)** Резултат: 5

Debug.Print Sqr(25)

В следващата таблица са представени най-използваните математически функции, вградени във **VBA.**

Функция	**Цел**
Abs (Число)	Абсолютна стойност
Cos(Ъгъл)	Косинус
Exp(Число)	Експоненциална функция
int(Число)	Цяло число
Log(Число)	Натурален логаритъм
Rnd(Число)	Рандомизация (случайно число)
Sin(Ъгъл)	Синус
Sqr(Число)	Квадратен корен
Tan(Ъгъл)	Тангенс

ФУНКЦИИ ЗА КОНВЕРТИРАНЕ

С тяхна помощ стойности от един тип данни могат да бъдат преобразувани в друг тип. Към тази група принадлежат функциите **str** и **val.**

Str(Число) - използва се при извеждане на символни низове и числа

Пример:

```
MsgBox (Ime+", ти си на "+str(Godini)+"години!")
```

След изпълнение на функцията ще се получи: Иво, ти си на 18 години!

В променливата Ime е съхранено името Иво, а в променливата Godini числото 18 Двете променливи са свързани с константен текст Тъй като целият израз трябва да бъде от един тип, функцията **str** преобразува числото 18 в символен низ - изписва се 18, но това вече не е число, а низ от два знака -1 и 8.

Val (Число) - използва се за преобразуване на знаков низ в число. При първия знак, който не е число, преобразуването се прекратява.

Пример: Val (символен низ)

Използва се в случаи, когато преобразувани в текст числа подлежат на обработка, т.е. с тях трябва да се извършват математически действия.

```
Dim Num
Num = Val("2345")' резултатът е числото 2345
```

Num = Val("1180 Sofia") 'резултатът е числото 1180

ТЕМА 12. ИТЕРАЦИИ (ПОВТОРЕНИЯ)

Понякога в хода на изпълнението на програмата се налага някоя част на програмата да се изпълни повече от един път, което води и новото изпълнение на блока от оператори в нея. В такива случаи всички езици за програмиране предлагат средства за организиране на цикли. **VBA** също предоставя на потребителя няколко възможности, а коя от тях ще бъде предпочетена, зависи преди всичко от конкретния проблем.

1. Do-Loop цикли

Програмни структури за организиране на циклично изпълнение на блок от оператори.

Намиращите се между ключовите думи **Do** и **Loop** оператори се изпълняват дотогава, докато не се изпълни условието за повторение. Проверката за изпълнение на условието може да се направи в началото, в края или някъде в тялото на цикъла.

Синтаксис 1:	Do [{While | Until} Условие] . . . Оператори **[If Условие Then Exit Do]** . . . **Loop**	Проверка на условието в началото на цикъла.Повторение на операторите от цикъла, докато условието е изпълнено **(while)** или докато условието не се изпълни **(until).** Възможност за преждевременно излизане от цикъла, ако се изпълни друго условие. След достигане на ключовата дума **Loop,** изпълнението отново се връща към оператора **Do.** Ако зададеното там условие още е в сила, операторите от тялото се изпълняват отново, в противен случай се излиза от цикъла. Управлението се предава на следващия оператор, който не принадлежи на цикъла

Синтаксис 2:	**Do** Оператори **[If** Условие **Then Exit Do]** **Loop [{while] Until}**Условие]	Възможност за преждевременно излизане от цикъла , ако е изпълнено даденото условие. Проверка на условието за повторение в края на цикъла.
	Забележка: При проверка на условието в края на цикъла, операторите от тялото на цикъла ще се изпълнят поне веднъж.	

2. While-Wend цикли

Програмна структура за организиране на циклично изпълнение на блок от оператори.

Намиращите се между ключовите думи while и `Wend` оператори ще се изпълняват дотогава, докато условието за повторение е изпълнено. Проверката на условието се прави в началото на цикъла. Тази структура наподобява структурата `Do whi le...Loop`.

Синтаксис:	**while** Условие Оператори **Wend**	Тук се проверява дали условието е изпълнено. При **Да** се изпълняват операторите от тялото на цикъла. При достигане на ключовата дума **wend,** програмата отново се отклонява към оператора **while**
	Забележка: За този вид организация на цикъл е характерно това, че проверката се извършва в началото на цикъла и е възможно цикъла да не се състои, ако условието не се изпълнява.	

3. For-Next цикли- *програмна структура за организиране на циклично изпълнение на блок от оператори*

Намиращите се между ключовите думи `For` и `Next` оператори ще се изпълняват дотогава, докато условието за повторение е изпълнено. Проверката на условието се прави в началото на цикъла или в неговото тяло.

Синтаксис:	**For** Брояч = Нач.значение **то** Кр.значение **[step** Стъпка] Оператори [If Условие **Then Exit For]** **Next** [Брояч] Броячът (идентификатор) на цикъла е числова променлива. При първото преминаване през цикъла, броячът получава начална стойност (напр. 1). Следват операторите от тялото на цикъла. Алтернативна възможност за преждевременно излизане от цикъла предлага командата **Exit For** при изпълнение на дадено условие. При достигане на оператор **Next** програмата се отклонява към началото на цикъла -оператор **For.** Броячът се увеличава според зададената стойност за стъпка (ако не е зададена такава стандартно се приема 1). Цикълът се изпълнява докато броячът достигне значението на крайната стойност.

Циклите могат да бъдат влагани един в друг. Един външен цикъл може да съдържа в себе си няколко вътрешни цикъла. На следващата схема е показан пример за вложени `For` - цикли. Тук е валидно следното правило: първият `For` принадлежи на последния `Next`, вторият `For` на предпоследния `Next` ит.н.

Пример

For I=1 **To** 100
Sum=Sum+1
Next I

Числовата променлива / се увеличава с 1 при всяко преминаване през цикъла. В тялото на цикъла всяка стойност на брояча се прибавя към Sum. След изработване на цикъла в Sum ще се натрупа сбора на числата от 1 до 100.

ТЕМА 13. ВЪВЕЖДАНЕ И ИЗВЕЖДАНЕ НА ДАННИ ЧРЕЗ ФУНКЦИИТЕ INPUTBOX, MSGBOX

1. **Функция InputBox -** много често в програмите се налага след въвеждане на данни в диалогово поле то да бъде затворено и ходът на програмата да продължи. **VBA** предлага на потребителя функцията **`InputBox`** с възможност за предварително дефиниране на диалоговото поле.

Синтаксис:	**inputBox**(Съобщение[,[Заглавие][,[Standart(|[,Xposition,YPosition]]])	
	Съобщение	Символен низ, който се извежда в диалоговата кутия с максимална дължина от 1024 знака. Ако съобщението е по-дълго от един ред ще трябва да се използват функциите **C h r** (13) за връщане на каретка и **c h r** (10) пред тази част на текста, която отива на другия ред.
	Заглавие	Символен низ, който се записва в заглавната ивица на диалоговата кутия. Ако такъв не се въведе, се изписва името на програмата.
	Standard	Символен низ, който да служи като стандартно значение.
	Xposition	Разстояние по хоризонтала между левия ръб на диалоговото поле и левия край на екрана.
	Yposition	Разстояние по вертикала от горния ръб на диалоговото поле до горния край на екрана.

След кликване на бутона ***OK*** от диалоговата кутия функцията **`InputBox`** връща на извикващата процедура символен низ (стринг) с въведените от потребителя данни в полето за въвеждане. Ако се клик-не на бутона ***Cancel*** се въвежда празен стринг ("").

2. Функция MsgBox - С помощта на тази функция потребителят получава междинни резултати в хода на изпълнението на програмата. Чрез кликване на бутон, той има възможност да реагира на съобщението в диалоговото поле. Работата с тази функция е много полезна за програмата, защото чрез отпечатване на данни и издаване на съобщения подпомага програмиста при откриване на грешки или при необходимост от решения на ситуации с отговори ***Да/Не.***

Синтаксис:

MsgBox(Съобщение [, Тип [.Заглавие]])

където: *Съобщение:* Текст, който ще се появи в диалоговото поле

Тип: Символен низ, който се появява в титулната лента на диалоговата кутия. Ако този параметър отсъства, се изписва името на съответната програма.

Заглавие: Символен низ, който се появява в заглавната ивица. Ако този параметър отсъства, се изписва името на съответната програма.