VBA (Visual Basic for Application)

1. ВЪВЕДЕНИЕ

1.1. Езици за програмиране

Развитието на компютрите, тяхната мощност, доведе до създаването на голям брой езици за програмиране, със своите особености и силно различаващи се един от друг.

Следните важни етапи са повлияли на развитието на езиците за прорамиране:

- създадване на Асемблер – преход от програмиране в кодове към символически адреси (1952г.);

- Създаване на машинно-независими езици от високо ниво: Фортран (Fortran, Formula Translator, 1956г.) за изчислителни задачи; Кобол (Cobol, Common Business Oriented Language, 1961г.) за решаване на комерчески задачи); PL/1 (Programing Language/One,1963-1966г.) универсален език за решаване на всякакви задачи, в това число и системни, които до този момент можеха да се решават само на Асемблер;

- Създаването на теорията на структурното програмиране доведе до появата на нови и и развитие на старите езици, реализиращи идеите на структурното програмиране. Най-ярък пример за такъв език е Паскал (1970г. създаден от Н. Вирт за обучение по програмиране)

Забележка: Основната идея на структурното програмиране е, че всеки алгоритъм може да бъде записан като композиция от три прости структури: последователност, разклонения, цикли.

Създаване на теорията на обектно-ориентираното програмиране, така както и структурното програмиране, повлия на създаването на езици за програмиране. Един от първите езици, реализиращи идеята на обектно-ориентираното програмиране е C++ (СИ++, 1983).

Забележка: Основната идея на обектно-ориентираното програмиране е обединение на данните и методите (подпрограмите) в едно цяло.

- Мощността на дадената методология е видима особено добре в езика VBA (1992г. Visual Basic for Aplication), създаден за разработване на приложения в средата на Microsoft Office.

- Други продукти, ползващи VBA могат да се видят на страницата:

http://msdn.microsoft.com/vba/companies/company.asp

Могат да се видят множество фирми, лицензирали VBA. Човек може да се замае от броя на приложенията, които могат да се обработват с VBA. Ето някои от тях:

1. COREL продукти – http://corel.com най-известните от които са WordPerfect и CorelDRAW.

2. Micrografx iGrafx серия - http://micrografx.com софтуер за създаване на диаграми, организационни графики и др.

3. IMSI TurboCad - http://turbocad.com софтуер за проектиране, тримерно моделиране, архитектурно проектиране, проектиране в машиностроенето, анимация и др.

1.2. История на създаване на VBA

VBA (Visual Basic for Application) – визуален обектно-ориентирн език за програмиране от високо ниво.

Важните етапи при създаването му са:

- 1964 г. се създава езика Basic (Beginners All-purpose Symbolic instruction Code) от Джон Кемени и Томас Куртс;

- 1990г. се появява графичната надстройка Windows 3 и Word for Windows. Във Word като макроезик се използва Basic;

- 1997г. се появява Microsoft Office 97, в който всички приложения изпозлват VBA.

1.3. За какво служи VBA?

· Автоматизирано създаване на документи, като текстови, електронни таблици и бази от данни

· Създаване на по-добър интерфейс на приложенията

· Изпълнение на изчисления по определени алгоритми

· Създаване на отчети и доклади от съществуващи бази от данни

· Добавяне на нови свойства /черти/ на едно приложение

· Създаване на специални средства, необходими при трансфер на документи – запазване на документи в алтернативни формати /XML eXtended Markup Language/

· Собствени настройки на начина на изпълнение на приложенията

1.4. Понятие за алгоритъм и програма

Алгоритъм – строго определена последователност от елементарни операции, изпълнението на които при определени начални данни /входни/ води до получаването на конкретни крайни резултати /изходни/.

Свойства на алгоритмите – определеност, крайност /финитност/, универсалност.

Алгоритъмът се явява коректен, ако при всички допустими начални стойности се получава верен резултат.

Пример: Алгоритъм на Евклид за намиране на НОД на 2 цели числа.

Средства за представяне на алгоритмите – естествен език, блок-схеми, език за програмиране. Ще представим алгоритъма на Евклид чрез горните средства.

1.5. Структурно програмиране – същност и основни принципи.

Структурното програмиране представлява определен стил за създаването на програмите. Спазвайки този стил, създадените програми придобиват прегледност, стават лесно читаеми и може автоматично да се верифицира алгоритъма. /верификация – доказателство за логическа коректност/.

Основните принципи на СП са:

- Програмите се съставят от блокове, всеки един от които има само един вход и един изход.
- Блоковете могат да се влагат един в друг, образувайки модули, като за всеки модул е в сила първият принцип.
- Всяка програма трябва да използва само три управляващи конструкции /блокове/ - последователност, разклонение /дихотомно, двоично/ и цикъл. Проф. Дейкстра е доказал, че с тези 3 конструкции може да се реализира произволен алгоритъм.

Всяка програма обикновено се състои от две части: описание на обектите (данните) и описание на действията, които трябва да се изпълнят над тези обекти.

Превеждането на алгоритъма на език разбираем за компютъра (например VBA) води до създаването на програма.

Програмата (подпрограма, функция) – представлява последователен набор от инструкции (описние на данни и действия), реализиращи алгоритъм за решаване на поставена задача чрез компютър.

2. Проект, модул и подпрограма – технология за програмиране във VBA

Езикът VBA е създаден за разработване на офисни приложения, като основни данни за него се явяват документите, създадени с офисните приложения на Microsoft Office. VBA позволява да се използват много възможности на Microsoft Office при работа с документите, но при това, програмите написани на езика VBA могат да бъдат изпълнени само в средата Microsoft Office.

Затова, при отваряне на нов документ, веднага се създава **проект**. Проектът включва модули, а модулите съдържат раздел Declaration (обявяване) и раздел подпрограми и функции.

2.1. Обектно-ориентирано програмиране – понятие за "обект".

Обектът от една програма прилича много на обектите от реалния свят. Програмистите са предложили този термин и неговото съдържание за да се създават лесно разбираеми и прегледни програми. Понятието обект от програмирането ще разясним с един пример от реалния свят – ябълката.

- **Свойства** на обектите /properties/ - могат да бъдат явни и скрити. Ако разгледаме ябълката като един обект, тя притежава явни и скрити свойства – цветът е едно явно свойство, вижда се /червена, жълта, зелена/. Вкусът, обаче е едно скрито свойство. Той се установява, когато отхапеш от ябълката. Обектът "**бутон**" от програмирането притежава заглавие /обяснение/ - caption, текстът, който потребителите могат да видят, гледайки бутона. Това е едно явно свойство. Обектите притежават и редица невидими /скрити/ свойства.

- **Метод** за обект /method/ - определя някакво действие върху обекта. Ако се върнем към примера с ябълката, висяща на дървото, тя може да бъде откъсната, като по този начин ще промени положението си в пространството. Бутонът, може да бъде преместен от едно място на друго с метод движение – *Move* method. Методите позволяват на програмистите да извършват определени действия върху обектите.

- **Събитие** /event/ - определена реакция, свързана с промяна в условията /състоянието/. Ябълката обикновено променя цвета си, когато зрее. Никой не може да направи нищо на ябълката - тя става зряла, защото достига своята зрялост. Това е едно събитие. Подобно на ябълката с VBA обектите могат да се случват събития. Потребителят кликва върху някой команден бутон и бутонът генерира *Click* event /събитие/. Събитията позволяват на програмистите да реагират в промяна на условията на обектите.

2.2. Специални обекти във VBA

Специалните обекти във VBA са три – форми, модули и клас модули. Формите съдържат потребителски интерфейсни елементи и служат за осигуряване взаимодействието с потребителя.

Модулите съдържат невидимия код на приложението, определени изчисления, описани в процедури и функции.

Клас модулите съдържат описанието на нови обекти, които желаем да създадем. Един клас модул може да служи за създаването на нов тип данни.

Тези три специални обекта могат да се разглеждат като контейнери за съхранение на програмен код и програмни елементи като клас дескриптори и процедури. Те се съхраняват обикновено заедно с приложението към което се създават, но могат да се съхранят и като самостоятелни файлове.

3. Основни елементи на езика за програмиране VBA

Език е знакова система, предназначена за съхраняване и предаване на информация. Както и за всички езици, така и за езиците за програмиране са валидни следните понятия:

- азбука – допустимите симоли на езика;
- лексика – начини за образуване на думите (лексемите) от символите (речник);
- синтаксис – метод на образуване на изреченията на езика;
- семантика – значението (смисъла на изреченията, тяхната логическа обвързаност);
- прагматика – предназначение и област на използване на езика.

Изхождайки от това, има смисъл да се даде описание на структурата на езика в следния ред: прагматика, лексика и синтаксис, и семантика.

3.1. Азбука на езика VBA

Азбуката на езика за програмиране е набор от символи, с помощта на които могат да бъдат образувани изреченията и операторите на даден език.

Символи – това е всичко, което може да бъде въведено от клавиатурата и от които се съставят лексемите на езика. Във VBA се използват всички

букви от латиницата и кирилицата, цифрите от 0 до 9 и препинателните знаци. Различните лексеми се изграждат от строго определени символи.

3.2. Лексеми

В таблица 3.1. са дадени лексемите на езика VBA и съответно символите, от които могат да бъдат изградени.

Таблица 3.1.

1	Число	Цифри от 0 до 9, символа ".", буквата E или e /латин/
2	Дата	**Цифрите от 0 до 9** Символът # Символът /????
3	Име (идентификатор)	**Латинските букви от A до Z и от a до z** **Кирилишки букви от А до Я и от а до я** Арабските цифри от 0 до 9 Символът подчертаване (долна черта) _ Символите @, #, &, $, %, !
4	Символен низ	**Всички символи на азбуката**
5	Разделители	**Символът интервал** Символът за продължение на реда – долна черта _ Символът табулация
6	Специални символи за построяване на различни конструкции	**+, -, *, `, ", ., <, >, =**

3.3. Правила за образуване на думите във VBA

Във VBA съществуват пет класа думи: имена; числа; дата; символен низ; ключови думи.

1. **Име** - последователност от символи (Табл. 3.1), съдържаща не повече от 255 знака, като първият символ трябва да бъде буква. Дадената последователност трябва да се отличава от ключовите думи. Главните и малки букви не се различават.

При работа в обвивката IDE, написването на име в програмата автоматично ще се приведе към първото написано име, т.е. ако сте въвели името MyName, а след това въведете, myname то myname ще бъде приведено към вида MyName. Тази възможност е полезна да се използва при набор на програмата, въвеждайки в името главна буква. Тогава по-нататък може да се набира това име с малка буква и ако името е набрано вярно, то в него ще се появи главната буква. Това значително улеснява писането на програми.

При съставянето на имената може да се спазват следните естествени правила:

- името да бъде кратко, но съдържателно;

- не е желателно да изпозлвате кирилица (ключовите думи са на английски и читаемостта на програмата се понжава, а ако на компютъра не се поддържа кирилица може да се получи грешка)

- Имената започват с малка буква, тъй като при ключовите думи първата буква автоматично се привежда в главна;

- при първа употреба на името в програмата задайте коментар (' текст на коментара);

- в случай на дълъг ред остатъкът се пренася на следващия ред като в края на предишния се поставя знакът долна черта "_";

2. **Число** – лексема, с помощта на която се записват числови стойности. Числовите значения могат да се запишат по три различни начина:

   - само с арабски цифри за записване на целите числа и знакът "+" или "-" преди числото за положителни или отрицателни стойности. Ако знакът се пропусне , то числото е положително;

   - с арабски цифри и точка, отделяща цялата от дробната част и съответния знак "+" или "-", например 3.141593, -1234.55;

   - с арабски цифри, точка, знакът "+" или "-" и символа E (e) от латиницата – предствяне на числото с плаваща десетична точка например 6.59E+5(65900), 1.342e-3(0.001342).

3. **Дата** – лексема за записване на дата

Допустими записи на дата:

#january 1 2006#

#jan 1 2006#

#1-jan-2006#

#1 1 06#

Датата трябва да се записва внимателно и да се разбере как датата се възприема от системата - например датата #1/5/99# във формата приет в USA е 5 януари 1999, а не 1 май 1999г.

4. **Символен низ** – произволна последователност от символи в двойни кавички. За включване на символа " в символния низ е необходимо той да се дублира.

5. **Ключова дума** – специална лексема, смисълът на която строго се регламентира от правилата на езика и служи за създадване на основни езикови конструкции. Не се допуска използването на ключовите думи за други цели.

4. ТИПОВЕ ДАННИ, КОНСТАНТИ И ПРОМЕНЛИВИ

4.1. Въведение

Всяка програма се състои от две части: описание на обектите, над които се изпълнява определен алгоритъм и описание на действията, които манипулират с тези обекти.

Обектите, с които оперира програмата, се наричат данни.

Данните имат два атрибута: име и стойност. Ако програмирахме в машинен код (асемблер), то бихме имали съответно адрес на клетка (име) и стойност – нейното съдържание, във вид на нули и единици. Езиците от високо ниво, позволяват да се абстрахираме от двоичното представяне на данните и да указваме на програмата, че двоичната информация в клетката е число, символен низ, дата и т. н., в зависимост от типа на данните, към които е отнесено името.

Тип на данните се нарича клас (група) обекти, стойността на които принадлежи на определно множество.

Типът на данните определя:

- Начина на представяне на данните в паметта на компютъра;
- областта на възможните стойности;
- допустимите операции за данните от този тип.

Типовете на данните се делят на прости (скаларни) и съставни (структурирани).

Прости типове данни – класове от обекти, всеки от които предствлява един елемент /данна/.

Съставни типове данни - класове от обекти, всеки от които включва няколко елемента /повече от един/.

4.2. Прости /скаларни/ типове данни

Във VBA са реализирани следните прости типове данни: логически, цели, дробни, дати, символни низове, обекти, вариантни и потребителски.

Първите осем типа данни са вградени, а деветият (потребителски) е тип данни, определен от програмиста. Всяка една от тези типове данни може да бъде константа или променлива.

4.2.1. Тип логически данни - Boolean

За проверка на различни условия и логически операции се въвеждат логически (булев) тип данни.

Логическият тип данни е клас обекти, които могат да приемат само две стойности: False и True. Константите от този тип се записват по същия начин.

False или True са ключови думи. В паметта на компютъра заемат 2 байта и приемат съответно стойности (0) или (-1).

За логическите променливи са допустими следните операции: not, and, or, xor, eqv, imp (Таблици 4.1. и 4.2.)

Таблица 4.1.

Операция	Описание
not a	Отрицание
a and b	Логическо И (конюнкция)
a or b	Логическо ИЛИ (дизюнкция)
a xor b	Изключващо ИЛИ/ednoto trqbva da e false
a eqv b	Еквивалентност/a=b
a imp b	Импликация/a=false ili b = true

Таблица 4.2.

a	b	Not a	a and b	A or b	a xor b	a eqv b	a imp b
True	True	False	True	True	False	True	True
True	False	False	False	True	True	False	False
False	True	True	False	True	True	False	True
False	False	True	False	False	False	True	True

Декларирането на променливи от този тип става по следния начин:

Dim MyBlnLog as Boolean

Пример за програма ...????????..

4.2.2. Числа

Във VBA целите и дробните числа имат повече от един представител:

- Byte – цяло без знак;
- Integer – цели числа;
- long – дълги цели числа;
- Single – дробно число;
- Double – дробно число с двойна точност;
- Currency – десетично число с фиксиран брой знаци след запетаята;
- Decimal – десетично число с фиксиран брой знаци след запетаята.

A. Данни от цял тип

Цял тип данни (Byte, Integer, Long) ....

B. Данни от дробен тип

Дробен тип данни (Single, Double, Currency, Decimal)

C. Данни от тип символен низ - String

D. Данни от тип дата - Date

E. Тип данни Object

F. Тип данни Variant

J. Потребителски тип данни

4.3. Константи и променливи

4.3.1. Константи

4.3.2. Променливи

5. Структура на VBA програма

За разлика от болшинството програмни езици, във VBA не съществува понятието главна програма. Структурата на VBA програмата включва два вида програмни единици – процедура (Sub) и процедура-функция (Function). Те са равностойни. В структурата ще включим също и определен вид дефиниции и компилаторски директиви. В записа на общия вид на операторите ще използваме квадратните скоби [ ] за

означаване на незадължителни параметри. Вертикалната линия -| ще използваме за разделяне на алтернативни параметри. Коментарните редове в една VBA програма се включват след символа апостроф (‘).

При направените уговорки, най-общо структурата на VBA изглежда така:

```
‘ Задаване на компилаторски директиви, ако е необходимо
[Option Explicit | Option Base | Option Compare | Option Private Module]
‘ Дефиниране на променливи и константи на ниво модул
[Public | Private име на променлива [as тип] ]
[Dim име на променлива [as тип] ] ‘ Dim прави същото както Private
‘ Дефиниране на константи на ниво модул
[Public | Private Const име на константа = стойност
‘ Дефиниране на процедура-функция
[Public | Private | Friend] [Static] Function име[(фиктивни параметри)] as тип
[оператори 1]
[име=израз1]
[Exit Function]
[оператори 2]
[име=израз2]
End Function
‘ Дефиниранре на процедура
[Public | Private | Friend] [Static] Sub име[(фиктивни параметри)]
[оператори]
End Sub
```

Компилаторските директиви указват начина на обработка на кода на програмата от компилатора на VBA. Те имат следното значение:

Option Explicit - указва явно деклариране на използваните в програмата величини. При наличие на тази директива компилатора извежда съобщение за грешка за всяка недекларирана величина. Препоръчително е задаването на тази директива с цел:

- Ускоряване изпълнението на кода
- Намаляване броя на грешките

Подобряване читаемостта на програмата

Option Base <число> - указва базата за индексиране на масивите. По премълчаване /липса на тази директива/ индексирането е от 0.

Option compare <метод> - определя един от възможните начини на сравняване на символни низове

Option Private Module - прави модула невидим за другите модули. Концепциите за private и public определят обсега на действие на обектите.

6. Оператор за присвояване, изрази и комуникация с приложението

6.1. Оператори за присвояване

Операторите за присвояване служат за промяна на стойността на променливите.

С помощта на операторите за присвояване се реализират конструкциите за последователност. Във VBA съществуват четири вида оператори за присвояване:

Let – обикновено присвояване;

LSet – ляво присвояване

RSet – дясно присвояване

Set – обектно присвояване.

6.1.1. Оператор Let (обикновено присояване)

Операторът служи за обикновено присвояване. Ключовата дума Let служи за съвместимост със старите програми на Basic. Тя не е задължителна. Общият вид на оператор за обикновено присвояванв във VBA е:

[Let] име на променлива=израз

където:

Let е ключова дума, която може да не се използва;

име на променлива – име на променливата, на която се присвоява стойност;

израз – изразът, даващ стойност на променливата, при което типът на получената стойност трябва да съответства на типа на променливата.

В този случай , когато променливата е от тип, определен от потребителя, е необходимо изразът да бъде бъде от същия тип.

6.1.2. Оператор Lset (присвояване с изравняване в ляво)

Използва се най-често при символни присвоявания. Например:

Lset име на променлива от символен тип=израз от символен тип

Може да се ползва и за копиране стойността на променлива от един потребителски тип в променлива от друг потребителски тип:

Lset име на променлива1 =име на променлива2

6.1.3. Оператор Rset (присвояване с изравняване в дясно)

Ползва се по същия начин както Lset, само че изравняването е в дясно

6.2. Изрази

Изразът представлява съвкупност от операнди (константи или променливи), знаци за операции и обръщение към функции.

Списък на аритметичните операции:

Знак	Синтаксис	Име/Описание
^	N1^N2	Степенуване/ Повдига N1 в степен N2
*	N1*N2	Умножение/Умножава N1 с N2
/	N1/N2	Деление/Дели N1 на N2
\	N1\N2	Целочислено деление/Дели N1 на N2 изрязвайки дробната част
Mod	N1 Mod N2	Остатък от целочислено деление/
+	N1 + N2	Събиране/ Събира N1 и N2
-	N1 – N2	Изваждане/Изважда N2 от N1

Логическите операции бяха разгледани в раздел 4.2.1. Трябва да се има предвид, че сравненията:

> - по-голямо; < - по-малко; >= - по-голямо или равно; <= - по-малко или равно; = - равно; <> - различно винаги водят до резултат от логически тип.

В изразите редът на изчисление се определя според приоритета на операциите, известен от математиката. При операции с еднакъв приоритет изчисленията се извършват отляво надясно. Редът на операциите може да бъде изменен с използване на скоби – ().

Знакът + се използва и за операцията конкатенация /слепване/ за символните типове.

Между отделните групи операции приоритетът е следния: аритметични, конкатенация на низовете, сравнения и логически.

Типът на резултата от изразите се определя в съответствие с някой правила, наречени приведени.

Приведено преобразуване – това е автоматично преобразуване на стойността от един тип в еквивалентна стойност в друг тип.

При автоматичното преобразуване може да се загуби информация, затова винаги е добре да се следи за преобразуването, като за целта се използват наличните във VBA функции.

6.3. Комуникация с приложението

Програмите, които ще съставяме с учебна цел, реализират алгоритми, които в по-голямата си част се нуждаят от въвеждане на данни и извеждане на резултати. За осигуряване на тези действия ще използваме стандартните функции на VBA – InputBox за въвеждане на данни и MsgBox за извеждане на резултати.

Общият вид на функцията е:

InputBox (съобщение [, заглавие на прозорец] [, стойност по премълчаване] [, x-позиция] [, y-позиция] [, help-файл, съдържание]) as String

Тя въвежда символен низ от диалогов прозорец със заглавие, ако такова е зададено като параметър и подканващ текст, който е единственият задължителен параметър в тази функция. Всички параметри, с изключение на първия са незадължителни.

Общият вид на функцията, която ще ползваме за извеждане на резултати е:

MsgBox (съобщение [, бутони] [, заглавие на прозорец] [, help-файл, съдържание]) as Integer

Функцията извежда съобщение в диалогов прозорец със заглавие, определено от третия параметър, ако е зададен. Ако във втория параметър са определени бутони, от стандартните във VBA, функцията връща резултат от тип Integer, съответстващ на натиснатия от потребителя бутон. Всички параметри, с изключение на първия са незадължителни. Ние най-често ще я използваме само с един параметър, представляващ извеждан резултат от програма на VBA.

7. Програмиране на линейни алгоритми

Това са най-простият тип конструкции, при които последователно се изпълняват входни операции, изчисления по определени изрази и присвояване стойността им на съответни променливи и извеждане на резултати. Като примери за такива алгоритми ще съставим своите първи програми на VBA.

Задача1. Съставете програма на VBA, която въвежда 3 числа и изчислява тяхната сума и произведение.

```
Option Explicit
Sub SumPr()
Dim a, b, c, Sum, Pr As Single
a = InputBox("Въведете число", "Въвеждане на данни", 5)
b = InputBox("Въведете число", "Въвеждане на данни", 8)
c = InputBox("Въведете число", "Въвеждане на данни", 10)
Sum = Val(a) + Val(b) + Val(c)
MsgBox "Сумата на числата е =" & Sum
Pr = a * b * c
MsgBox "Произведението на числата е =" & Pr
End Sub
```

8. Програмиране на разклонени алгоритми

VBA предоставя голямо оразнообразие от оператори за разклонение. Тук ще разгледаме само най-често използваните в програмите оператори за разклонение.

8.1. Кратка форма на оператор за разклонение *If … Then*

Тази форма на оператора за разклонение съществува в две разновидности – редова структура и блочна структура. Ще разгледаме и двете разновидности, които са напълно заменяеми.

8.1.1. Редова структура на кратката форма

Общ вид:

If логически израз Then оператор1 : оператор2 : ... операторN

Логическият израз се пресмята и в зависимост от резултата се предприемат следните действия:

- ако резултатът има стойност True се изпълняват операторите след Then. Техният брой не е ограничен, но трябва да се намират в същия ред. След това се изпълнява оператора от следващия ред в програмата.

- ако резултатът има стойност False се изпълнява направо оператора от следващия ред в програмата. Операторите след Then се прескачат.

8.1.2. Блочна структура на кратката форма

Общ вид:

```
If Логически израз Then
Оператор1
```

Оператор2
...
ОператорN
End If

Изпълнението е аналогично на редовата форма. Първо се пресмята логическият израз и в зависимост от резултата се предприемат действията:

- ако резултатът има стойност True се изпълняват операторите между Then и End If и след това се изпълнява оператора след End If в програмата

- ако резултатът има стойност False се изпълнява направо оператора след End If.

8.2. Пълна форма на оператора за разклонение If … Then … Else

Както при кратката форма и тук съществуват две разновидности. Едната е с редова структура, а другата с блочна.

8.2.1. Редова структура на пълната форма

Общ вид:

If Логически израз Then Оператор1:Оператор2: ... ОператорN Else Оператор1:Операtр2: ... ОператорN

Целият оператор трябва да се записва в един ред. Изпълнението и на този оператор започва с пресмятането на логическия израз и след това в зависимост от неговия резултат се предприемат действията:

- ако резултатът има стойност True се изпълняват операторите след Then и след това оператора от следващия ред в програмата

- ако резултатът има стойност False се изпълняват операторите след Else и след това оператора от следващия ред в програмата.

8.2.2. Блочна структура на пълната форма

Общ вид:

If Логически израз Then
Оператор1
Оператор2
...
ОператорN
Else
Оператор1
Оператор2
...

```
ОператорN
End If
```

Изпълнението отново започва с пресмятането на логическия израз и в зависимост от неговия резултат се предприемат действията:

- ако резултатът има стойност True се изпълняват операторите между Then и Else и след това оператора след End If

- ако разултатът има стойност False се изпъляват опертаторите между Else и End If и след това оператора след End If.

Използването на тези оператори ще демонстрираме със следните задачи:

Задача 1. Съставете програма на VBA, която въвежда координати на точка в равнината – x,y и определя в кой квадрант лежи точката.

```
Sub tochka()
' Въвеждане координатите на точка в равнината
x = InputBox("Въведи x координата ", "Въвеждане на данни")
y = InputBox("Въведи y координата ", "Въвеждане на данни")
  If x * y > 0 Then 'ако произведението е положително точката е в 1 или 3 квадрант
    If x > 0 Then
       MsgBox "Точката лежи в квадрант 1"
            Else
       MsgBox "Точката лежи в квадрант 3"
    End If
            Else ' в случай на отрицателно произведение точката е във 2 или 4 квадрант
    If y > 0 Then
       MsgBox "Точката лежи в квадрант 2"
            Else
       MsgBox "Точката лежи в квадрант 4"
    End If
  End If
End Sub
```

Задача 2. Съставете програма на VBA, която въвежда аргумента x и изчислява стойността на функцията:

$$Y = \begin{cases} x-1, & за\ x<0 \\ x^2-2x+1, & за\ x\geq 0 \end{cases}$$

Ще използваме пълната форма на оператора If с редова структура:

```
Option Explicit
Sub funkcia()
Dim x, y As Single
' Въвеждане на аргумента x
x = InputBox("Въведи аргумент", "Въвеждане на данни", 1)
```

```
If x < 0 Then y = x - 1 Else y = x ^ 2 -2*x+ 1
MsgBox "Стойност на функцията =" & y
End Sub
```

Задача 3. Съставете програма на VBA, която въвежда аргумента x и изчислява стойността на функцията:

$$Y = \begin{cases} x-1, & за\ x<0 \\ x^2+1, & за\ 0 \le x<1 \\ 2x^2-3x+1, & за\ x \ge 1 \end{cases}$$

Най-подходяща за този тип задачи, където функцията е дефинирана в повече от два интервала е кратката форма с редова структура:

```
Option Explicit
Sub funk10()
Dim x, y As Single
' Въвеждане на аргумента x
x = InputBox("Въведи аргумент", "Въвеждане на данни", 1)
If x < 0 Then y = x - 1
If x >= 0 And x < 1 Then y = x * x + 1
If x >= 1 Then y = 2 * x ^ 2 - 3 * x + 1
MsgBox "Стойност на фумкцията =" & y
End Sub
```

8.3. Оператор за разклонение в повече от две посоки - Select Case

9. Програмиране на циклични алгоритми

Многократното повторение на едни и същи действия за различни стойности на определени параметри ще наричаме цикъл. *Във VBA са реализирани различни форми на оператори за цикъл.* Многообразието на цикли позволява създаването на програми с подходяща реализация на циклична организация за различни случаи.

9.1. Цикъл с предварително зададен брой повторения – For .. Next

От предоставените два варианта на цикъл с предварително зададен брой повторения ще разгледаме само първия, който представлява най-простата циклична конструкция и е достатъчен за нашите цели.

Общ вид:

For Брояч = начална стойност To крайна стойност [Step размер на стъпката]

```
Оператор 1
Оператор2
...
```

ОператорN
Next [Брояч]

Където брояч е произволна променлива от числов тип, началната и крайна стойност са изрази от числов тип. *Когато стъптата не е зададена явно, се подразбира по премълчаване 1. Всяка следваща стойност на брояча ще бъде с 1 по-голяма от предходната. Размер на стъпката, когато е зададена ключовата дума Step е също числов израз. Тогава нарастването е с изчислената стойност на този израз. Поставянето на Брояч след ключовата дума Next е полезно от гледна точка на прегледност /лесна читаемост на програмата/ при вложени цикли*.

Ще използваме цикъла For … Next за реализиране на класическите алгоритми за натрупване на сума и произведение.

Задача 1. Съставете програма на VBA за определяне на сумата от първите n естествени числа – Sum=1+2+ … + n = ∑i , (i=1,2, … n).

Когато тябва да получим сума, независимо от участващите в тази сума числа, трябва предварително да занулим /да присвоим 0 на/ определена за целта променлива. В болшинството съвременни компилатори това се прави при стартиране на програмата за всички променливи, но с учебна цел, ние винаги ще записваме в нашите програми оператор за зануляване на определената за сума променлива. След това, в цикъл ще прибавяме нова стойност към съществуващата в клетката, избрана за сума.

```
Sub suma()
Dim n, i, sum As Integer
n = InputBox("Въведете цяло число", "Въвеждане на данни")
sum = 0
For i=1 to n
sum = sum + i
Next
MsgBox "Сумата от първите " & n & " числа има стойност" & sum
End Sub
```

Задача 2. Съставете програма на VBA за определяне на произведението от първите n естествени числа.

В математиката произведението 1.2.3 ... (n-1).n се отбелязва с n! и се нарича факториел.

Програмата на ще се отличава от предната само по инициализацията на променливата, определена за натрупване на произведението и по операцията *, която ще замени +. Променливата, определена за натрупване на произведение винаги ще зареждаме предварително със стойност 1.

```
 Sub Proizwedenie()
Dim n, i, Pr As Integer
n = InputBox("Въведете цяло число", "Въвеждане на данни")
Pr  = 1
For i=1 to n
Pr = Pr * i
Next
MsgBox "Произведението от първите " & n & " числа има стойност" & Pr
End Sub
```

Стъпката в цикъла може да бъде и отрицателна. Тогава имаме цикъл с намаляващ брояч.

9.2. Цикли с поставено условие за край - Do … Loop

Съществуват четири конструкции на цикли с условие за край. Те се различават по мястото на поставяне на условието за край /предусловие и постусловие/ *и стойността на логическия израз, при която се прекратява /продължава/ изпълнението на тялото на цилъла – While Until.*

9.2.1. Цикъл с предусловие – Do While … Loop

Общ вид:

```
Do While логически израз
Оператор1
Оператор2
...
ОператорN
Loop
```

Изпълнението на този оператор протича по следния начин: Първо се пресмята стойността на логическия израз след While. Ако неговата стойност е True се изпълняват операторите след Do While и отново се изчислява стойността на логическия израз. Ако стойността на логическия израз е False се прекратява цикъла – изпълнява се оператора след Loop в програмата. От казаното за начина на изпълнение на този цикъл, може да се направят следните изводи: *Тялото на цикъла може да не бъде изпълнено нито един път.* За да се осигури възможност за прекратяване на цикъла след краен интервал от време трябва да съществува поне една променлива, участваща в логическия израз, която променя стойността си в тялото на цикъла /поне веднъж се среща от лявата страна на оператора за присвояване/.

9.2.2. Цикъл с предусловие – Do Until … Loop

Общ вид:

```
Do Until логически израз
Оператор1
Оператор2
```

...
ОператорN
Loop

Разликата между цикъла в т. 9.2.1. и този цикъл е само в стойността на логическия израз при която се прекратява /продължава/ цикъла. При цикъла Do Until ако логическият израз има стойност False се изпълняват операторите от тялото на цикъла и отново се изчислява логическия израз, а при стойност True се пркратява цикъла. Всичко останало, казано за начина на изпълнение и принципа, посочващ начин за проверка коректността на условието за край са напълно идентични.

9.2.3. Цикъл с постусловие – Do … Loop While

Общ вид:

Do
Оператор1
Оператор2
...
ОператорN
Loop While Логически израз

След като се изпълнят операторите от тялото на цикъла се изчислява стойността на логическия израз. Ако тя е True се изпълняват отново операторите от тялото на цикъла. Когато стойността на логическия израз стане False цикълът се прекратява. Характерна особенност на този тип цикли е че тялото на цикъла се изпълнява поне веднъж, без значение каква е стойността на логическия израз.

9.2.4. Цикъл с постусловие – Do … Loop Until

Общ вид:

Do
Оператор1
Оператор2
...
ОператорN
Loop Until логически израз

Разликата между този цикъл и предходния е само в стойността на логическия израз, при която се прекратява /продължава/ цикъла. Стойността на логическия израз, при която се продължава изпълнението на цикъла е ,а тази при която се прекратява е.

За да илюстрираме неудобството при използване на този тип цикли при решаване на задачата от т. 9.1. ще съставим програмата за намиране на сумата от първите естествени числе прилагайки цикъл с предусловие.

Sub suma()

```
Dim n, i, sum As Integer
n = InputBox("Въведете цяло число:", "Въвеждане на данни")
i = 1
sum = 0
Do While i <= n
sum = sum + i
i = i + 1
Loop
MsgBox "Стойността на сумата е=" & sum
End Sub
```

Както се вижда, освен допълнителната инициализация на променливата i, тук се налага и да променяме сами нейната стойност в тялото на цикъла. Този тип цикли са подходящи за задачи, при които повторението на определени действия се извършва до настъпване на някакво събитие. Типичен пример за това е алгоритъма на Евклид, който разгледахме още в началото на нашите занимания. В него се извършваше актуализация на първоначално въведените две цели числа, докато те станат равни. Ето как ще изглежда програмата, реализираща алгоритъма на Евклид, ползваща цикъл с предусловие.

```
Sub Euclid()
Dim a, b As Integer
a = InputBox("Въведете първото цяло число", "Въвеждане на данни", 5)
b = InputBox("Въведете второто цяло число", "Въвеждане на данни", 3)
Do Until a = b
If a > b Then a = a – b Else b = b - a
Loop
MsgBox "Най-големият общ делител има стойност" & a
End Sub
```

9.3. Прекратяване изпълнението на макрос или процедура.

Когато изпълняваме програми с циклични конструкции има вероятност да се случи така нареченото зацикляне. Вече споменахме за възможността при цикли с предварително определено условие за край да се случи така, че веднъж получена стойност на логическия израз, тя да остане същата поради това, че в тялото на цикъла не се среща нито един оператор, в който се променя стойността на поне една променлива, участваща в логическия израз. В такива случаи е необходимо да се прекрати изпълнението на макроса външно, тъй като програмата сама никога няма да спре. За прекратяване изпълнението на програма трябва да се натисне клавиша Esc или комбинацията Ctrl+Break. VBA завършва изпълнението на текущия оператор и преминава в състояние на очакване, при което се извежда диалогов прозорец Code execution has been interrupted с четири бутона: Continue; End; Debug; Help.

Натискането на бутона Continue продължава изпълнението на програмата.

Натискането на бутона End завършва изпълнението на програмата, при което се губи стойността на всички променливи.

Натискането на бутона Debug активира режим Break на VBA, при който може да се проследи стойността на определени променливи, като се изпълнят ръчно няколко стъпки от цикъла.

Възможността за Help не носи никаква нова информация, освен, че програмата е била спряна в резултат от натискането на Esc или Ctrl+Break.

10. Съставни типове данни

Във VBA са реализирани два типа съставни данни:

- масиви;
- тип, определен от потребителя.

10.1. Масиви

Масивите са първият съставен тип данни, който се появява в езиците за програмиране от високо ниво, и се използва в изчислителните задачи при работа с вектори и матрици. В литреатурата се наричат подреден тип данни.

Масивът е клас от обекти, които се състоят от елементи от един и същи тип /наредена последователност от еднотипни елементи, означавани с едно и също име, снабдено с един или повече индекси/.

Във VBA са реализирани два типа масиви – статични и динамични масиви. *В много езици за програмиране терминът масив се използва за статичните масиви, тъй като в тях не се реализират динамични* (СИ, Паскал и др.).

10.1.1. Статични масиви

Ако броят на елементите е известен предварително се изпозлват статични масиви.

Статичният масив е съвкупност, състояща се от фиксиран брой елементи.

Синтаксисът при дефиниране на масиви е:

Dim | Private | Public | Static име на масив ([индекси]) [As тип]

където:

индексите, ако се задават представляват следната последователност:

[долна граница1 To] горна граница1 [, [долна граница2 To] горна граница2] ... [, [долна границаN To] горна границаN]

Както се вижда единствените задължителни параметри в дефиницията са името на масива и кръглите скоби след него. Когато е зададена само горната граница за индекс, долната се приема по премълчаване и е определена от компилаторската опция Option Base. Когато не е зададен типът на базовите елементи се приема по премълчаване за Variant. Когато и Option Base не е зададен номерацията се приема от 0.

Dim | Private | Public | Static, To, As са ключови думи.

Опцията Option Base в модула предшества описанието на подпрограмата и функцията и действа в границите на дадения модул.

За масивите не са разрешени никакви операции, а за елементите на масивите са възможни тези оперции, които са допустими за типа данни към които се отнасят.

Посочването на елемент от масив /цитиране/ става, като след името в кръгли скоби се поставят действителни индекси.

Например петивтелемент от едномерния масив с име A ще се цитира с A(5). Действителните индекси могат да бъдат константи, променливи или изрази.

10.1.2. Работа с едномерни масиви

В този раздел ще илюстрираме ползването на масиви в програмите на VBA с множество характерни алгоритми върху едномерни масиви.

Задача 1. Съставете програма на VBA, която въвежда елементите на едномерния масив A(M), M ≤ 30 и определя сумата от неговите елементи.

```
Option Explicit
Option Base 1
Sub sumarray()
Dim i, m As Integer
Dim Sum As Single
Dim A(30) As Single
m = InputBox("Въведете брой елементи на масива", "Въвеждане на данни", 5)
' Логически контрол на текущия брой елементи
Do While m < 1 Or m > 30
m = InputBox("Грешен брой елементи. Въведи ново m", "Въвеждане на данни", 5)
Loop
For i = 1 To m
A(i) = InputBox("A[" & i & "]=", "Въвеждане елементите на масива A", 1)
Next
Sum = 0
For i = 1 To m
Sum = Sum + A(i)
```

```
Next
MsgBox "Стойността на сумата от елементите на масива е=" & Sum
End Sub
```

Задача 2. Съставете програма на VBA, която въвежда елементите на едномерния масив А(м), м≤40 и определя най-големия елемент и неговото място в масива.

```
Option Explicit
Option Base 1
Sub maxelement()
Dim i, m, kmax As Integer
Dim Max As Single
Dim A(40) As Single
m = InputBox("Въведете брой елементи на масива", "Въвеждане на данни", 5)
' Логически контрол на текущия брой елементи
Do While m < 1 Or m > 40
m = InputBox("Грешен брой елементи. Въведи ново m", "Въвеждане на данни", 5)
Loop
For i = 1 To m
A(i) = InputBox("A[" & i & "]=", "Въвеждане елементите на масива A", 1)
Next
Max = A(1)
kmax = 1
For i = 2 To m
If A(i) > Max Then Max = A(i): kmax = i
Next
MsgBox "Стойността на най-големия елемент от масива е=" & Max
MsgBox "Най-големият елемент от масива се намира на " & kmax & " място"
End Sub
```

10.1.3. Динамични масиви

В случаите, когато размерността на масив не е известна се използват динамични масиви.

Динамичният масив е масив, при който броят на елементите и техния тип може да се променя в процеса на работа на програмата.

Синтаксис:

Dim | Private | Public nameArray () [As]

където:

nameArray е името на масива;

nameType е името на типа на елементите;

Dim | Private | Public, As са ключови думи.

За динамичните масиви не са разрешени никакви операции, а за елементите на масивите са възможни тези операции, които с допустими за типовете данни, към които те се отнасят.

За работа с динамичните масиви, заделяне на памет за тях (определяне на размерността, границите на изменение на индексите, типът на елементите) или промяна (промена на размерността, границите на изменение на индексите, типът на елементите) се изпозлва операторът Redum.

10.2. Типове данни, определени от потребителя

Както за простите типове данни е въведен потребителския тип Enum, така и за съставните е дадена възможност за създадване на собствени типове данни.

Типът данни, определен от потребителя в другите езици за програмиране често се нарича записи. Затова този термин ще се изпозлва за дадения тип данни.

Типът на данните, определн от потребителя (записи) е клас обекти, съдържащи фиксиран брой елементи. *Елементите могат да бъдат от различен тип.* Синтаксисът е:

За записите са допустими само операции присвояване, а за елементите операции, които са допустими за типовете данни, към които те се отнасят.

11. Област на действие на имената. Ключови думи – Dim, Public, Private, Static

Във VBA се създадва проект . Проектът съдържа модули, а модулите – подпрограми и функции.

Променливите и константите могат да се обявяват както на ниво модул (до описанието на подпрограми и функции), *така и на ниво подпрограми* (след оператора Sub) *и функции* (след оператора Function). *В зависимост от това, къде е обявена променливата* (константата) *и с каква ключова дума Dim, Public, Private или Static (Public или Pravite) с нея може да се работи или само в подпрограмата (функцията) или на ниво модул (във всички подпрограми и функции на дадения модул), или на ниво проект (променливата ще бъде достъпна във всички подпрограми и функции на всички модули). Това свойство се нарича област на видимост на променливата (константата).*

На ниво **модул** може да бъде използвана всяка форма на обявяване на променлива (освен Static) и константа. *При това, ако обявяването е Private или Dim (за променливата), то дадената променлива или*

константа е достъпна във всички процедури или функции. Трябва да се отбележи, че ако в подпрограмата (функцията) е обявена променлива или константа със същото име, както и на ниво модул, то тя скрива тази, обявена на ниво модул, т. е не може да се получи информация за обявената променлива (константа) на ниво модул.

Ако променливата или константата е обявена на ниво модул като Public, то тя ще бъде видима във всички модули на проекта. При това, ако в друг модул (подпрограма, функция) е описана променлива или константа със същото име, то за работа с променливата или константата, описана с ключовата дума Public, е необходимо да се използва следващия запис:

nameModul.nameVar, където:

nameModul е името на модула в който константата или променливата е обявена като Public;

nameVar – име на променливата или константата.

На ниво *подпрограма* и *функция* при обявяването на променливите се използват само ключовите думи Dim или Static, а константите се определят без ключовите думи Public или Private. *Променливите и константите, обявени на ниво подпрограма (функция) са видими само в нея. В случай, че променливата е обявена с ключовата дума Static, то стойността на дадената променлива се съхранява и след изхода от подпрограмата (функцията), но за другите подпрограми и функции тя ще бъде недостъпна, и при следващото влизане в дадената подпрограма ще може да се използва стойността ѝ.*

11.1. Префикси в имената на VBA

Препоръчва се имената да се използват с два префикса. Първият ще инфромира прогрмиста за областта на видимост на данните (Табл. 3.5.), а вторият за типа данни (табл. 3.6.).

Таблица 3.5.

Префикс	Описание
G	Global за Public
M	Module за Private
отсъствие	за променливи дефинирани в подпрограмата (функцията)

Таблица 3.6.

Префикс	Тип на данните
Str	String
Int	Integer
Byt	Byte
Ing	Long
Sng	Single
Dbl	Double
Cur	Currency
Var	Variant
Obj	Object
Bln	Boolean