

ЛЕКЦИЯ 2

ПРЕГЛЕД НА ЕЗИКА ВБ

- ⌚ Запис на програмата
- ⌚ Идентификатори и именование
- ⌚ Типове данни и структури
- ⌚ Дефиниране на величини
- ⌚ Операции и изрази
- ⌚ Оператори
- ⌚ Процедури и функции
- ⌚ Условна компилация

ВБ 2

1 / 64

ОБЩИ БЕЛЕЖКИ

В **началото на 60-те** години на ХХ век за да се облекчи приобщаването на новаци към програмирането на компютри е създаден **прост ЕПВР**, наречен **Basic** – **Beginner's All purpose Symbolic Instruction Code** (**символен код от инструкции с общо предназначение за новобранци**).

Езикът е **изключително неудачен** и скоро **бива отречен** от специалистите, защото **води до много** програмни **грешки** и **не отговаря** на новосъздадените изисквания към ЕПВР, известни като „**Структурно програмиране**“.

ВБ 2

2 / 64

BASIC И МАЙКРОСОФТ

Езикът **Basic** се възражда при появата на **ПК**, а фирма **Майкрософт** започва своето съществуване с печалбите от интерпретатори на **Basic** за **ПК**.

Създаването на удобни средства за програмиране при **ОС** с **ГПИ**, кара **МС** да приведе езика **Basic** към съвременните изисквания на професионалните програмисти: **структурно и ОО програмиране**.

За да не пострадат **стари клиенти**, създали някакъв програмен код на **Basic**, **новосъздаденият** език **ВБ** е **съвместим с класиката**, макар че се препоръчва да се използват само новите, разумни решения. **ВБ** е млад, но **постоянно се развива и усъвършенства**.

ВБ 2

3 / 64

ПРАВИЛА ЗА ЗАПИС

- ❶ **Начинът за писане** на буквите **не** оказва **влияние** на написаното, т. е. **Б ≡ б**.
- ❷ **Прост оператор** трябва да се запише изцяло **в** рамките на **един** логически **ред**.
- ❸ Последователни **физически редове** могат да бъдат обединени **в един логически ред** **чрез** знака за **подчертаване** () в края си.
- ❹ **Някои части** на структурните оператори трябва да са **сами в** рамките на **един ред**.

ВБ 2

4 / 64

ЗАПИС (продължение)

- ⑤ **Операторите**, записани в един ред, се разделят с двоеточие (:).
- ⑥ **Етикет** може да има **само първия** оператор на всеки **ред**, защото **етикетите се отделят** от операторите **също с двоеточие** (:).
- ⑦ В **началото на ред** може да се запише **цяло число**, което е негов **номер** и допълнителен **етикет на първия оператор** в него.
- ⑧ Записаното **от знака апостроф (')** до края на логическия ред **е коментар**.

ВБ 2

5 / 64

ИДЕНТИФИКАТОРИ

Започват с буква, продължават с букви, цифри и знак за подчертаване (), и не могат да бъдат служебни думи на езика.

Кирилицата е равноправна на латиницата.

Когато именуват **програмни елементи** броят на знаците е ограничен на **255**.

Когато именуват **обекти** (**форми**, **свойства**, **елементи** на ГПИ) броят на знаците е **40**.

След името на величина и функция може да бъде записан **знак за определяне на типа**.

ВБ 2

6 / 64

ВИДИМОСТ НА ИМЕНАТА

- ❶ Имената на величините, декларирани в една процедура (функция), са локални за тази процедура (функция).
- ❷ Имената на величините, декларирани в общата част на модул, са глобални на равнището на този модул.
- ❸ Имената на процедурите в модул-форма са глобални на модулно равнище за нея.
- ❹ Имената на процедурите в стандартен модул са глобални за цялата програма.

ВБ 2

7 / 64

ВИДИМОСТ НА ИМЕНАТА (прод.)

- ❺ Имената на елементите на ГПИ, носени от дадена форма, са глобални на модулно равнище за тази форма.
- ❻ Имената на формите са глобални за цялата програма.
- ❼ При конфликт на имена се използва най-близкото определение, т. е. локалните имена на процедурно ниво закриват глобалните имена, а глобалните имена на модулно равнище закриват имената на глобалните елементи на програмата.

ВБ 2

8 / 64

ИМЕНОВАНЕ

- ❶ Имената на видимите в един участък програмни елементи се цитират **директно**.
- ❷ Имената на елементите на ГПИ, носени от една форма, се цитират **в нея директно**.
- ❸ Имената на методите и свойствата на формите могат да се цитират **директно** чрез името си **в тази форма**: **<име>**.
- ❹ Имена, методи, свойства и елементи на ГПИ на чужда форма се цитират като **пред тях** се запише **името на формата** и точка (.): **<форма>.<име>**.

ВБ 2

9 / 64

ИМЕНОВАНЕ (прод.)

- ❺ Свойствата на елементи на ГПИ се цитират **във формата, която ги носи**, добавени с точка (.) **към името на съответния елемент**: **<елемент>.<свойство>**.
- ❻ Всеки клас от обекти (вкл. и елементите на ГПИ) **има свойство**, което се **подразбира** в случай, че **след името** на някой екземпляр на класа (или след обектова променлива, указател към екземпляр от този клас) **не бъде записано** никакво **свойство**.
- ❼ Методите на елементите на ГПИ се цитират **като свойства**: добавени с точка (.) към името им – **<елемент>.<метод>**.

ВБ 2

10 / 64

ИМЕНОВАНИЕ (прод. 2)

- ⑧ **Свойства и методи** на елемент от друга форма се цитират като **пред тях** се пише нейното **име**, последвано от точка (.):
<форма>.<елемент>.<свойство | метод>.
- ⑨ **Процедурите** на една форма могат да се използват и **в други форми** като **пред тях** се запише **името на дефиниращата форма** и точка (.): **<форма>.<процедура>.**
- ⑩ **Текущият екземпляр на** дадена форма може да се цитира в нейния код и чрез стандартната именована константа **Me**.

ВБ 2

11 / 64

ИМЕНОВАНИЕ (прод. 3)

- ① ① **Имената на събитийните процедури** на елементите на ГПИ са **предопределени**. Те се образуват като **към името** на елемента се допише знак за **подчертаване** (_) и името на събитието: **<елемент>_Click.**
- ① ② **Предопределените имена на събитийните процедури във всяка форма** са еднакви и имат вида **Form_<събитие>: Form_Load.**
- ① ③ **Имената на формалните параметри** на събитийните процедури се изписват наготово от текстовия редактор, защото също са **предопределени**.

ВБ 2

12 / 64

ВРЕМЕ НА ЖИВОТ

Локалните променливи на процедура **се създават при нейното стартиране и се унищожават при нейното завършване** (**изход** от нея **с Exit или End**).

Глобалните променливи на модулно равнище за формите се създават при въвеждане на формата в паметта и се унищожават при нейното извеждане от паметта.

Глобалните променливи на програмата се създават при нейното стартиране и не се унищожават до нейното завършване.

Всички променливи се раждат с определена начална стойност: 0, "", Empty или Nothing!

ВБ 2

13 / 64

ТИПОВЕ ДАННИ

❶ цяло число (**INTEGER** в Паскал):

Byte: $0 \div 255$, 1 байт ОП, няма знак;

Integer: $-32\,768 \div 32\,767$, 2 байта, знак %;

Long: $-2\,147\,483\,648 \div 2\,147\,483\,647$, 4 байта, знак &.

❷ приближено число (**REAL** в Паскал):

Single: 0 и абс. с-т $1,401298 \times 10^{-45} \div 3,402823 \times 10^{+38}$, 4 байта ОП, знак !;

Double: 0 и абс. стойност от $4,94065645841247 \times 10^{-324}$ до $1,79769313486232 \times 10^{+308}$, 8 байта, знак #.

❸ логически (**BOOLEAN** в Паскал):

Boolean: True (–1) и False (0), 2 байта ОП, няма знак.

ВБ 2

14 / 64

ТИПОВЕ ДАННИ (прод.)

④ знаков низ (**липсва** в Паскал):

String – променлива дължина до около 2 милиарда,
10+брой знаци байта ОП, знак \$;

String * n – фиксирана дължина n до 65 400 знака,
брой знаци байта ОП, няма знак;

литералните константи се заграждат със знак за инч ("");

⑤ указател към (**липсва** в Паскал) – 4 байта ОП:

име на клас: екземпляр от този определен клас;

Control: (екземпляр на) произволен елемент на ГПИ;

Form: (екземпляр на) произволна форма;

Collection: екземпляр на специалния клас колекция;

Object: екземпляр от съвсем произволен клас;

Nothing е литерална константа за невалиден указател.

Променливите от тези типове **се наричат обектови** и се ползват за достъп до екземпляри на посочения клас.

ВБ 2

15 / 64

НОВИ ТИПОВЕ ДАННИ

⑥ парична сума – **Currency**:

мащабирано цяло число в 8 байта, знак @ и диапазон
от -922 337 203 685 477,5808
до +922 337 203 685 477,5807;

⑦ дата (и час) – **Date**:

приблизено число Double (8 байта), чиято **цяла част**
определя **датата** като брой дни спрямо 30 декември
1899 г., а **дробната част** – **времето** от полунощ;

представими са датите от 1 януари 100 г. до 31
декември 9999 г., но без 30.12.1899 г.

литералните константи се заграждат със знака диез (#)
и **стандартът** е **#мм-дд-гггг чч:мм:сс [С/В]#** като С/В е
АМ за сутрин или РМ за след обед: #1 октомври
2002#, #12:30#, #3:30 РМ#.

ВБ 2

16 / 64

УНИВЕРСАЛЕН ТИП

⑧ Нарича се **Variant** и е подразбиран във ВБ.

Стойността може да бъде **от всеки тип** без String*n и още:

Липсваща (**Empty**);

Неопределена (**Null**);

Грешка (**Error**);

произволен **масив**;

произволен **запис**;

от десетичен тип **Decimal**, който **не съществува самостоятелно** – 12-байтово двоично цяло без знак, представящо 29-цифрено десетично число с естествена запетая.

Числовите стойности заемат **16 байта**, знаковите низове – броя на знаците в тях + **22 байта**.

Променливите от този тип **се маскират** на променливи от **типа на своята текуща стойност**, включително на масив и запис. Този текущ тип може да се проверява.

ВБ 2

17 / 64

СТРУКТУРИ ОТ ДАННИ

ВБ позволява работа с **два вида структури** от данни: **масиви и** (логически) **записи**.

Масивите се определят също като простите променливи и **не са отделен тип**.

Размерността им не може да бъде над **60**.

Масивите биват **два вида**: с фиксиран размер (**статични**) и с променлив (**динамични**).

Декларацията на фиксиран масив определя неговите **размерност, гранични двойки и тип на елементите**.

ВБ 2

18 / 64

СТАТИЧНИ МАСИВИ

- ❶ Граничните двойки по всяка размерност на фиксиран масив се определят разделени със запетая (,) в кръгли скоби (()) след името на масива във вида [**<начало> To <край>** (**<начало> ≤ <край>!**).
- ❷ Липсата на **начална стойност** я установява по **подразбиране** на **0** или на **1**, когато в общата част на модул има оператор **Option Base 1**.
- ❸ Параметрите на фиксираните масиви **не могат да се променят**.

ВБ 2

19 / 64

ДИНАМИЧНИ МАСИВИ

- ❶ Параметрите на динамичните масиви не се определят при декларирането им (но в него се пишат скобите) и трябва да бъдат определени (или променени) **с изпълнение на оператор Redim**.
- ❷ Динамичен масив може да участва и в лявата страна на оператор за **присвояване**. Такъв **може да бъде и резултатът от изпълнение на функция**.
- ❸ Произволен **масив** може да бъде **стойност на променлива от тип Variant**.

ВБ 2

20 / 64

РАБОТА С МАСИВИ

- ❶ **Елемент на масив** се цитира като след името му в кръгли скоби (**()**) и разделени със запетая (**,**) бъдат записани произволни изрази: **Масив(1,2)**.
- ❷ **Особен случай** с два комплекта скоби е цитиране на елемент на масив, който се явява стойност на елемент на друг масив, но вече от тип **Variant**: **МВ(1)(2,3), ДМВ(5,7)(3)**.
- ❸ ВБ предлага следните **стандартни функции**:
 - **LBound(<име>[,<размерност>])** – долната граница за <размерност> (**1**);
 - **UBound(<име>[,<размерност>])** – горната граница за <размерност> (**1**);
 - **Array(<списък>)** – генерира едномерен масив.

ВБ 2

21 / 64

РАБОТА С ДИНАМИЧНИ МАСИВИ

Dim <име>() – деклариране;

ReDim [Preserve] <име>(<гр. двойки>) [As <тип>] – **определя (променя) параметрите** на масива:

- **типът** на елементите може да **се променя** само **когато един масив е стойност на променлива** от тип Variant;
- **без Preserve старите** стойности на масива **се губят**;
- с **Preserve** може да се променя **само горната граница по последната размерност** и стойностите **се запазват**;
- при неизвестно име **ReDim** е еквивалент на **Dim**.

Erase <име> – **освобождава памет** при динамичен масив и **нулира елементите** на статичен масив.

ВБ 2

22 / 64

(ЛОГИЧЕСКИ) ЗАПИСИ

- ❶ Структурата на записите (имената и типовете на техните полета) има статут на отделен тип данни, наречен потребителски.
- ❷ Такава структура може да се определя само в общата част на модул без етикети и номерация на редовете чрез следния запис:


```
[Private | Public] Type <име на макет> ◀
  <име на поле 1>[[<индексни двойки>]] [As <тип>] ◀
  [<име на поле 2>[[<индексни двойки>]] [As <тип>] ◀
  ...
End Type ◀
```
- ❸ След подобна декларация <име на макет> може да се използва като потребителски тип данни.

ВБ 2

23 / 64

ЗАПИСИ (продължение)

- ❹ Декларацията на потребителски тип не създава променливи, явяващи се структура от данни (логически) запис, а само осигурява възможност за деклариране на подобни променливи.
- ❺ Във ВБ е разрешено стойността на една променлива от потребителски тип да бъде присвоявана директно на друга променлива от същия потребителски тип.
- ❻ Записите могат да бъдат и резултат от изпълнението на потребителски функции.
- ❼ Запис може да бъде и стойност на променлива от тип Variant, при което тя временно има статут на структура от данни (логически) запис.

ВБ 2

24 / 64

ИЗПОЛЗВАНЕ НА ЗАПИСИ

Индивидуално поле от променлива, явяваща се логически запис, се посочва като **след името на променливата се запише точка (.)** и след нея **името на желаното поле**:
Запис.Поле.

Името на **променливата** може да **не се пише** (но **не и точката пред** името на **полето**)
в блок от вида:

```
With <променлива> ◀  
    [ оператори ] ' .Поле  
End With ◀
```

ВБ 2

25 / 64

БЛОК WITH

Блокът **With ... End With** осигурява и **директен достъп до свойствата и методите на екземпляр** от определен клас, когато името на този екземпляр се запише вместо <променлива>, явяваща се структура от тип запис.

Блоковете **With могат да се влагат** един в друг, стига поле на запис (свойство) само по себе си да се явява запис.

ВБ 2

26 / 64

МЕХАНИЗМИ ЗА СЪДАВАНЕ НА НОВИ ТИПОВЕ ОТ ДАННИ

Практическата насоченост на ВБ прави **ненужен** присъщия за Паскал механизъм **поддиапазон**.

Поради своята голямата практическа приложимост в програмирането във ВБ е **осигурен механизъмът за изброяване**.

Езиковият процесор предоставя **наготово редица изброими типове**.

ВБ 2

27 / 64

ИЗБРОЕН ТИП

Нови типове на изброяване се създават с конструкцията:

```
[Public | Private] Enum <име на тип> ◀  
  <Име 1> [= <константен израз 1>] ◀  
  <Име 2> [= <константен израз 2>] ◀  
  ...  
End Enum ◀
```

Числовите стойности на литералните константи <Име i> по премълчаване са **последователни** и започват **от 0**.

Те не могат да се променят в програмата.

ВБ 2

28 / 64

ИМЕНОВАНИ КОНСТАНТИ

Системата за проектиране **ВБ** предоставя **наготово стотици именовани константи**.

Техният брой се **увеличава**, когато към даден проект бъде присъединена специфична **обектна библиотека**, създадена при дефинирането на нов клас от обекти.

Дефинирането на собствени именовани константи става чрез записи от вида:

[Public | Private] Const <име> [As <тип>] [= <израз>]

ВБ 2

29 / 64

ИМЕНОВАНИ КОНСТАНТИ (прод.)

При определянето на именована константа:

- ❶ В **<израз>** могат да участват **само литерални и вече определени именовани константи**;
- ❷ **Public** и **Private** не могат да **се използват на процедурно равнище и в общата част на модул-форма**;
- ❸ **Public** в общата част на **стандартен модул** означава, че константата е достъпна във **всяка част на програмата**;
- ❹ **Private** в общата част на **стандартен модул** означава, че константата е достъпна **само в рамките на този модул**.

ВБ 2

30 / 64

ДЕФИНИРАНЕ НА ПРОМЕНЛИВИ

Променливите могат да бъдат дефинирани по два начина: **чрез явна декларация и неявно.**

Неявно дефиниране на променлива се осъществява **само на процедурно равнище чрез записване на неизвестно до този момент (невидимо в този диапазон) име.**

Неявното дефиниране на променливи е изключително опасно!

Оператор **Option Explicit** в общата част на модул **забранява неявно дефиниране** в него.

ВБ 2

31 / 64

НЕЯВНО ОПРЕДЕЛЯНЕ НА ТИП

При неявно дефиниране и при липса на **тип** в явно дефиниране на променлива този тип **се определя:**

❶ **по специфичен знак**, записан **след името** (идеология на **класическия Basic**, валидна само за определени типове от данни);

❷ **по първата буква на името** (идеология на **Фортран**) при условие, че в общата част на съответния модул присъства запис от вида: **DefXXXX** <диапазон 1> [, <диапазон 2> . . .], където **XXXX** е съкращение на типа, а <диапазон i> се определя чрез <начална буква>[-<крайна буква>];

❸ като **универсален (Variant)** в останалите случаи.

ВБ 2

32 / 64

ЯВНО ДЕФИНИРАНЕ

Явното дефиниране се осъществява чрез служебните думи **Dim**, **Public**, **Private** и **Static**.

Public и **Private** могат да се използват **само** в общата част на **стандартен модул** и определят **видимостта на името**:

Public – в цялата програма;

Private – само в съответния модул.

Static може да се използва **само на процедурно равнище** и определя **време на живот** равно на времето за **изпълнение на програмата**, т. е. подобни **променливи запазват своята текуща стойност** между две поредни изпълнения на съответната процедура.

ВБ 2

33 / 64

ЯВНО ДЕФИНИРАНЕ НА ПРОМЕНЛИВИ

Явно дефиниране на променлива може да се осъществи **на всяко място** от програмата, но **до използване** на това име, чрез запис от вида:
Dim <име>[(< гр. двойки >)] [**As** [**New**] <тип>] [, ...]
 Всички променливи се „раждат“ с начална стойност:

0 при всички **числови типове**;

празен низ ("") при тип **String**;

n знака Nul (0) при тип **String*n**;

Empty (**Липсваща**) при тип **Variant**.

Стойността на **обектовите променливи** е **Nothing** **без New** или **указател към създадения в момента нов екземпляр** на класа **при наличие на New**.

ВБ 2

34 / 64

АРИТМЕТИЧНИ ОПЕРАЦИИ

Прилагат се към операнди от числов тип:

- ^ – повдигане на степен;
- * – умножение;
- / – делене с резултат обикновено Double;
- \ – целочислено делене с предварително **отрязване на дробната** част на операндите;
- Mod** – намиране на остатък от целочислено делене с предварително **закръгляване** на операндите до цяло число;
- + – събиране;
- – смяна на знака и изваждане.

ВБ 2

35 / 64

СРАВНЕНИЯ

- ① <операнд 1> <знак> <операнд 2>
 - типът на операндите **определя** и **начина** на **сравняване** като **числа** или като **знакови низове**;
 - **резултатът** е от **логически** тип (**True** = -1, **False** = 0);
 - <знак> е < за по-малко, <= за по-малко или равно, > за по-голямо, >= за по-голямо или равно, = за равенство и <> за различие.
- ② <низ> **Like** <образец> – съпоставяне с образец, в който ? = произволен единичен знак, * = произволен брой знаци (вкл. 0), # = една цифра (0 до 9), [<знаци>] = съвпадение с един от посочените знаци (а-в = а, б и в), [!<знаци>] = несъвпадение с всички посочени знаци.

ВБ 2

36 / 64

СРАВНЕНИЯ (прод.)

- ③ <указател 1> **Is** <указател 2> е истина само когато и двата указателя сочат към един и същи екземпляр на обект;
- ④ **TypeOf** <указател> **Is** <име на клас> е истина само когато екземплярът, който сочи <указател>, е от посочения клас.
- ⑤ Сравняването на низове се определя от оператор **Option Compare { Binary | Text | Database }** в общата част на модула и по подразбиране е **Binary** = по код, а **Text** = независимост от вида на буквата (Б=б).

ВБ 2

37 / 64

ОПЕРАЦИИ С НИЗОВЕ

- ① **Единствената операция е конкатенация (слепване)** на низове със знак **&**. Знак **+** също може да означава конкатенация при определени обстоятелства.
- ② Като стандартни функции са реализирани редица допълнителни операции с низове като **извличане** на **подниз** (**Left, Right, Mid**), **търсене** на подниз (**InStr, InStrRev**), определяне на **броя на знаците** (**Len**), **преобразуване** (**StrConv, LCase, UCase, LTrim, RTrim, Trim, StrReverse, Replace**), **генериране** (**Space, String**), **разбор** (**Split**) и др.
- ③ За **промяна на подниз** се използват специалните оператори **Mid, LSet** и **RSet**.

ВБ 2

38 / 64

ЛОГИЧЕСКИ ОПЕРАЦИИ

Прилагат се към операнди от логически тип:

Not – отрицание (инверсия);

And – конюнкция (логическо И);

Or – дизюнкция (логическо ИЛИ);

Xor – сума по модул 2 (изключващо ИЛИ);

Eqv – еквивалентност;

Imp – импликация.

Логическите операции реализират тризначна (**True**, **False** и **Null**) вместо двузначна логика.

Приложени към числови операнди **същите имена** реализират аналогичните **побитови операции**.

ВБ 2

39 / 64

СТОЙНОСТ NULL

Може да бъде **текуща стойност само на** променливи от универсален тип (**Variant**).

Всяка операция, освен логическите, чийто операнд е **Null** води до резултат също **Null**.

Това означава, че сравнението **PrVar = Null** винаги ще даде резултат **Null**, който ще бъде трансформиран в Лъжа (**False**).

За да се реши този проблем ВБ предоставя логическата функция **IsNull(PrVar)**, която връща стойност Истина единствено при текуща стойност на **PrVar Null** (**Неизвестна**).

ВБ 2

40 / 64

ТРИЗНАЧНА ЛОГИКА

Логическите операции не размножават стойност **Null** тъй като вместо обичайната двузначна (**Булева**) логика прилагат логика с три стойности: **Истина** (**True**), **Лъжа** (**False**) и **Неопределено** (**Null**).

Таблиците на истинност са:

x	Not x
True	False
False	True
Null	Null

ВБ 2

41 / 64

ТАБЛИЦИ НА ИСТИННОСТ

x	T	T	T	F	F	F	N	N	N
y	T	F	N	T	F	N	T	F	N
x And y	T	F	N	F	F	F	N	F	N
x Or y	T	T	T	T	F	N	T	N	N
x Xor y	F	T	N	T	F	N	N	N	N
x Eqv y	T	F	N	F	T	N	N	N	N
x Imp y	T	F	N	T	T	T	T	N	N

ВБ 2

42 / 64

ИЗРАЗИ

При изчисляване на израз операциите имат следния **приоритет по намаляване**:
степенуване (^), смяна на знак (унарн –),
умножение и делене (*, /), целочислено
делене (\), остатък от делене (Mod),
събиране и изваждане (+, –), конкатенация
(&), всички сравнения с еднакъв приоритет
(=, <>, <, >, <=, >=, Like, Is), отрицание (Not),
конюнкция (And – И), дизюнкция (Or – ИЛИ),
сума по модул 2 (Xor – изключващо ИЛИ),
еквивалентност (Eqv), импликация (Imp).
Операциите, които са **равни по приоритет**,
се изчисляват **от ляво на дясно**.

ВБ 2

43 / 64

ИЗРАЗИ (прод.)

Изчисляването на един израз при ВБ се подчинява на някои необичайни правила:

- ❶ Когато **резултатът** от дадена операция е вън от разрешения за операндите диапазон става **автоматично повишаване** към по-широк тип:
Byte → **Integer** → **Long** → **Double**.
- ❷ При **необходимост типът** на операнд **може да бъде преобразуван** към друг, стига това преобразуване да осигури възможност за нормално изпълнение на операцията: **String** може да бъде събран с число, стига неговата текуща стойност да е правилен запис на число.
- ❸ Универсалният тип има **специални стойности**, които директно се преобразуват към друг тип: **Empty** се преобразува в **0** или **""**.

ВБ 2

44 / 64

ПРЕОБРАЗУВАНЕ НА ТИП

За принудителното **преобразуване на типа** на даден израз ВБ предоставя следните **стандартни функции**:

CBool(X) – към **Boolean**; **CByte(X)** – към **Byte**;
CCur(X) – към **Currency**; **CDate(X)** – към **Date**;
CDbl(X) – към **Double**; **CDec(X)** – към **Decimal**;
CInt(X) – към **Integer**; **CLng(X)** – към **Long**;
CSng(X) – към **Single**; **CStr(X)** – към **String**;
CVar(X) – към **Variant**.

ВБ 2

45 / 64

ПРОВЕРКА НА СТОЙНОСТ

За **проверка на стойността** на израз от тип **String** и **Variant** ВБ предоставя **две функции** с резултат от **логически тип**:

IsDate(x) – валиден запис на дата и
IsNumeric(x) – валиден запис на число.

За проверка на **текущата стойност на променлива** от тип **Variant** има три функции:

IsArray(<пром.>) – **масив**;
IsEmpty(<пром.>) – **Липсваща (Empty)** и
IsNull(<пром.>) – **Неопределена (Null)**.

ВБ 2

46 / 64

СТОЙНОСТИ НА VARIANT

Типът на текущата стойност на променлива **Variant** може да бъде изследван и със стандартна функция **VarType(<пром.>)**, която връща като резултат :

vbEmpty (0), **vbNull** (1), **vbInteger** (2), **vbLong** (3),
vbSingle (4), **vbDouble** (5), **vbCurrency** (6),
vbDate (7), **vbString** (8), **vbObject** (9), **vbError** (10),
vbBoolean (11), **vbDecimal** (14), **vbByte** (17) или
vbUserDefinedType (36).

Когато текущата стойност е **масив** с елементи от съответния тип горните стойности **се увеличават с vbArray** (8192).

При **масив** с елементи **Variant** резултатът е:
vbArray (8192) + **vbVariant** (12).

ВБ 2

47 / 64

ПРОВЕРКА НА ТИП

Освен функция **VarType**, приложима главно за променливи **Variant**, може да се използва и функция **TypeName**, чийто **резултат** от тип **знаков низ** е: **"Byte"**, **"Integer"**, **"Long"**, **"Single"**, **"Double"**, **"Currency"**, **"Decimal"**, **"Date"**, **"String"**, **"Boolean"**, **"Error"**, **"Empty"**, **"Null"**, **"Unknown"** – неизвестен тип, **"Object"** – OLE обект, **"<име на клас>"** – обектова променлива, която сочи екземпляр от този клас, и **"Nothing"** – обектова променлива, която не сочи никакъв екземпляр.

При **масиви** **резултатът** е **"Variant()"** или името на типа на елемента **с добавени скоби** – **"Byte()"**.

Записи не могат да се проверяват с TypeName.

ВБ 2

48 / 64

ФУНКЦИИ С АЛТЕРНАТИВЕН РЕЗУЛТАТ

ВБ предоставя няколко **стандартни функции**, чиито **резултат** се определя едва **по време на изпълнение** на програмата. Такива са:

👉 **условната функция:**

If(<условие>, <стойност ДА>, <стойност НЕ>)

✌ **алтернативната функция:**

Choose(<израз>, <стойност 1> [, <стойност 2> ...]), при която <израз> се окръглява до цяло.

Choose връща **Null** при **некоректна стойност** на <израз> (<1 или >броя на стойностите).

ВБ 2

49 / 64

ПРОСТИ ОПЕРАТОРИ

👉 оператори за **присвояване** (4 броя):

[Let] <променлива> = <израз>

Set <об. променлива> = <об. израз>

LSet <пром.> = <знаков израз/пром.>

RSet <пром.> = <знаков низ>

✌ оператори за **безусловен преход** (5):

явен: GoTo <етикет | номер на ред>

неявни: Exit ... (Do, For, Sub, Function)

👉 оператори за **работа с форми** (2):

Load <форма> – въвеждане в ОП;

Unload <форма> – извеждане от ОП.

ВБ 2

50 / 64

УСЛОВНИ ОПЕРАТОРИ

Редови (съвместимост с Basic):

If <усл.> **Then** <оп. И> [**Else** <оп. Л>] ◀

Блоков (структурен):

If <условие 1> **Then** ◀

[<оператори 1>]

[**Elseif** <условие 2> **Then** ◀

[<оператори 2>]

. . .]

[**Else** ◀

[<допълнителни оператори>]

End If ◀

ВБ 2

51 / 64

ОПЕРАТОР ЗА ИЗБОР

Select Case <тестов израз> ◀

Case <списък 1> ◀

[<оператори 1>]

[**Case** <списък n> ◀

[<оператори n>] . . .]

[**Case Else** ◀

[<допълнителни оператори>]]

End Select ◀

<Списък *i*> са разделени със запетая (,):

<израз>, <начало> **To** <край> или

[**Is**] <знак за сравнение> <израз>

ВБ 2

52 / 64

ЦИКЛИ С УСЛОВИЯ

С предусловие:

```
Do { While | Until } <условие> ◀
  <оператори>      ' Exit Do !
Loop ◀
```

```
While <условие> ◀
  <оператори>
Wend ◀
```

Със следусловие:

```
Do ◀
  <оператори>      ' Exit Do !
Loop { While | Until } <условие> ◀
```

ВБ 2

53 / 64

ДРУГИ ЦИКЛИ

С преброяване (аритметична прогресия):

```
For <пром.> = <нач.> To <край> [Step <стъпка>] ◀
  <оператори>      ' Exit For ◀!
Next [<пром.>] ◀
```

```
For Each <об. пром.> In <колекция/група> ◀
  <оператори>      ' Exit For ◀!
Next [<обектова променлива>] ◀
```

Безкраен:

```
Do ◀
  <оператори>      ' Exit Do ◀!?!
Loop ◀
```

ВБ 2

54 / 64

ДЕФИНИРАНЕ НА ПРОЦЕДУРИ И ФУНКЦИИ

Процедура:

```
Sub <име> (<форм. параметри>) ◀
    <оператори>          ' Exit Sub !
End Sub ◀
```

Функция:

```
Function <име> (<форм. парам.>) [As <тип>]
    <оператори>          ' Exit Function !
End Function ◀
```

Пред **Sub/Function** може да се запише **Private** | **Public** за видимост и **Static** за живота на локалните променливи.

ВБ 2

55 / 64

ДЕКЛАРИРАНЕ НА ПОДПРОГРАМИ

ВБ дава възможност за използване на **подпрограми**, разположени в библиотека за динамично свързване (**DLL**). За да се използва подобна подпрограма тя трябва да бъде **декларирана** в общата част на модул:

```
[Public | Private] Declare {Sub | Function} <ВБ име>
    Lib "<библиотека>" [Alias "<истинско име или
    #номер>"] [(<формални параметри>)] [As <тип>]
```

Win32api.txt съдържа **готови декларации** на подпрограмите в ядрото на Windows.

Използването на подпрограми става чрез <ВБ име> **по подобие** на процедурите и функциите **на ВБ**.

ВБ 2

56 / 64

ФОРМАЛНИ ПАРАМЕТРИ

Списъкът с формални параметри съдържа разделени със запетая (,) описания от вида:

[**ByVal** | **ByRef**] <име> [()] [**As** <тип>]

ByVal – предаване на параметъра **по стойност**;

ByRef – предаване **по позоваване** (**подразбира се**);

() – параметърът е **масив**;

<тип> – подразбира се тип **Variant**.

Само при деклариране с **Declare** на готова подпрограма в **DLL** като тип може да се укаже и **Any** за посочване на произволен тип.

ВБ 2

57 / 64

ПАРАМЕТРИ ПО ИЗБОР

ВБ позволява създаване на **процедури** (функции) **с параметри**, които не са задължителни при използването им, т. е. параметрите са **по избор**.

Пред името на незадължителен параметър се записва **Optional**, а **след** определяне на **типа му** може с константа да се посочи и **стойността му по подразбиране**: **Optional** <име> [= <стойност>] .

Всички формални параметри, посочени **след първия незадължителен**, **също** трябва да бъдат **по избор**!

Стандартна функция **IsMissing**(<име>) в дефиниция на процедура **проверява дали нейното активиране е било със или без фактически параметър** за <име>.

ВБ 2

58 / 64

ПРОИЗВОЛЕН БРОЙ ПАРАМЕТРИ

ВБ позволява и създаване на **процедури с произволен брой параметри**. Те не могат да имат незадължителни параметри.

За целта пред името на **последния** формален **параметър** се записва **ParamArray**, което означава, че той е масив с елементи **Variant** за допълнителните фактически параметри.

За този последен параметър не може да се посочва **ByVal** и **ByRef**, и не трябва да има изборни (**Optional**) формални параметри.

ВБ 2

59 / 64

ИЗПОЛЗВАНЕ НА ПРОЦЕДУРИ И ФУНКЦИИ

Процедурите се активират по два начина:

 **<име>** <фактически параметри>

 **Call** **<име>**(<фактически параметри>)

Функциите се използват само в изрази, където се заместват с изчислената от тях стойност (**последната присвоена на името им** или 0, "", **Empty, Nothing**) чрез

<име>(<фактически параметри>)

ВБ, подобно на Си, има право да **промени реда за изчисляване** на изразите, поради което **не са желани странични ефекти**.

ВБ 2

60 / 64

ИМЕНОВАНИ ПАРАМЕТРИ

Обичайното съответствие между фактически и формални параметри **е по позиция**.

Освен него, **ВБ позволява** и използване на **ключови (именовани) фактически параметри**.

Всички параметри на **дефинираните** в програмата процедури (функции) и много от параметрите на стандартните за ВБ процедури, функции и методи **са ключови**.

При поименно съответствие фактическите параметри се записват **в произволен ред**, но заедно с имената си: **<име> := <стойност>**

ВБ 2

61 / 64

УСЛОВНА КОМПИЛАЦИЯ

За да се облекчи създаването на съвкупност от сходни програми ВБ осигурява **апарат за условна компиляция**, чрез който **част от програмата** се трансформира **в коментар**:

#If <константно условие 1> **Then** ◀

<текст 1>

[#Elseif <константно условие 2> **Then** ◀

[<текст 2>] ...]

[#Else ◀

[<допълнителен текст>]]

#End If ◀

ВБ 2

62 / 64

УПРАВЛЕНИЕ НА КОМПИЛАЦИЯТА

В **константните изрази** могат да се използват само **литерални константи**, **константи за управление на условната компилация** и **знаците за операции без Is**.

На **модулно равнище** константи за условна компилация се създават чрез

#Const <име> = <константен израз> ◀

Глобални константи на програмно равнище се създават чрез средата за разработка.

VBA осигурява две глобални **константи**, описващи средата на работа: **Win16** и **Win32**.

ВБ 2

63 / 64

**БЛАГОДАРЯ ВИ
ЗА ВНИМАНИЕТО!**

**БЪДЕТЕ С МЕН И
В СЛЕДВАЩАТА ЛЕКЦИЯ,
КОЯТО ЩЕ НИ ОТВЕДЕ
В НЕВЕРОЯТНИЯ СВЯТ НА
ОБЕКТИТЕ НА ГПИ**