

ЛЕКЦИЯ 1

ОСНОВНИ ПОНЯТИЯ

- ⌚ Принципи на процедурното програмиране
- ⌚ Принципи на обектното програмиране
- ⌚ Принципи на събитийното програмиране
- ⌚ Какво е Визуъл Бейсик?
- ⌚ Структура на програмата при ВБ

ВБ_1

1 / 20

ПРОЦЕДУРНО ПРОГРАМИРАНЕ

При традиционните езици като Паскал:

- ① Самата програма изцяло **определя** коя нейна част и кога ще бъде изпълнена;
- ② Изпълнението започва от началото и **следва** **предопределен път**, който е гъвкав, тъй като зависи и от началните данни;
- ③ Прилага се методът на **най-ранно време за свързване** между елементите на езика и техните конкретни характеристики.

ВБ_1

2 / 20

ПРОБЛЕМИ ПРИ КЛАСИЧЕСКИТЕ ЕЗИЦИ

- ❶ Родени са по време на **пакетната еднопрограмна обработка**.
- ❷ Входните данни се четат **от карти**.
- ❸ Изходът е **печатащо** устройство.
- ❹ Взаимодействието със **запомнящите устройства** е **слабо развито**.
- ❺ Липсват средства за **паралелно изпълнение**.

ВБ_1

3 / 20

ОБЕКТНО ПРОГРАМИРАНЕ

Еволюция на идеите при програмиране:

- ❶ **Структурно** програмиране – **яснота** на записа;
- ❷ **Модулно** програмиране – **разделно създаване** на модули от програмите със:
 - ➔ **статично** обединяване на модулите от свързващ редактор;
 - ➔ **динамично** обединяване на модулите от ОС.
- ❸ **Обектно** програмиране – **обобщение** на принципа „разделяй и владей“:
 - ➔ **клас** от обекти – укрива се реализацията им;
 - ➔ **екземпляр** от даден клас – използване на класа.

ВБ_1

4 / 20

ЕЛЕМЕНТИ НА ОБЕКТИТЕ

Всеки клас притежава характерни за него:

 **СВОЙСТВА** - осигуряват възможност за **управлението и** сведения за **състоянието** на конкретен **екземпляр**;

 **МЕТОДИ** - осигуряват възможност за **въздействие** върху **и използване на** конкретен **екземпляр**.

Всеки екземпляр е достъпен **чрез указател**, който се получава при създаването му, и е невалиден след разрушаването му.

Обслужването на един **клас** от обекти може да бъде **осигурявано от** (е грижа и дело на):

- ➔ **езиковия процесор** при статично свързване;
- ➔ **ОС** при динамично свързване чрез сървър на класа.

ВБ_1

5 / 20

СЪБИТИЙНО ПРОГРАМИРАНЕ

При събитийното програмиране **програмата е в аморфно (изчакващо) състояние** и **настъпването на** определени **събития определя кога и коя част** от програмния код **ще бъде изпълнена**.

В този вид програмиране **изпълнението** на програмата **зависи от събитията**, които ще възникнат.

Реакцията на определено **събитие чрез** изпълнение на програмен **код не е задължителна**. Реакцията на събитие, за което липсва програмен код, е стандартна, съгласно общоприетите разбирания на ОС.

При събитийното програмиране **класовете от обекти** имат **допълнителна** характеристика: **набор от събития**, на които техните екземпляри имат право да реагират.

ВБ_1

6 / 20

ИНИЦИАТОРИ НА СЪБИТИЯ

Инициатор (генератор) на едно **събитие** може да бъде:

- ❶ извършване на определено **действие** от страна **на лицето**, което използва (**изпълнява**) програмата – натискане на **клавиш**, придвижване или щракване на **мишката** и др.
- ❷ **ОС** – оповестяване, че текущият сеанс на работа с компютъра завършва и др.
- ❸ **езиковия процесор** – изтичане на зададен период от време.
- ❹ **програмният код** – изпълнение на специфични **оператори** или прилагане на определени **методи** към даден екземпляр на обект.

ВБ_1

7 / 20

ОСОБЕНОСТИ

Събитията възникват асинхронно, следователно:

- ➔ **докато** в отговор на дадено събитие **се изпълнява** програмен **код за реакцията** му **може** да възникне **друго** (в частност, **и същото** събитие);
- ➔ **програмните участъци**, отговорни **за реакцията** на събития, по естествен начин **се изпълняват паралелно**;
- ➔ **някои събития са взаимно свързани**, т. е. едното не може да възникне без преди това да се е случило другото – **двукратното** щракване с мишката **се предшества от еднократно**, **разпознаването** на стандартен клавиш **се предшества от натискането** му и др.

ВБ_1

8 / 20

ПРЕДИМСТВА

Програма, управлявана от събития, е:

- ➔ по-пригодна за **диалогово** изпълнение;
- ➔ по-приспособена към изискванията на съвременния **ГПИ**;
- ➔ пригодена за **паралелно изпълнение** по естествен начин;
- ➔ **по-удобна за създаване**, защото често въобще не е необходимо писане на програмен код.

ВБ_1

9 / 20

КАКВО Е ВИЖУЪЛ БЕЙСИК?

- ❶ интегрирана **система за проектиране** (**създаване**) **на програми**, управлявани от събития, включваща:
 - ❖ текстов **редактор**;
 - ❖ **транслатор** от смесен или компилативен тип;
 - ❖ програма за проверяване и поправяне (**дебъгер**);
 - ❖ **рисуване** на елементите на ГПИ, вместо тяхното текстово записване (**Visual** = **зрителен, нагледен**).
- ❷ **език за програмиране**, чрез който се описва реакцията на възникващите събития.

ВБ_1

10 / 20

РЕАЛИЗАЦИИ НА ВБ

- ❶ За приложения (**VBA** – Visual Basic **for Application**) – част от всеки продукт на MS Office след версия 97;
 - ❷ Учебна (**VB Learning** Edition);
 - ❸ Професионална (**VB Professional** Edition);
 - ❹ Промислена (**VB Enterprise** Edition).
- Различават се само по класовете от обекти, които се получават наготово заедно със средата за проектиране.

ВБ_1

11 / 20

ВЕРСИИ НА ВБ

- 1.0 (1991 г.) – псевдографика при MS DOS.
- 2.0 (1992 г.) – за Windows 2.0.
- 3.0 (1993 г.) – за Windows 3.0, I-ва стабилна.
- 4.0 (1995 г.) – произвежда 16-битов код за Windows 3.0 и 32-битов за Windows 95.
- 5.0 (1997 г.) – превод до машинен език и създаване на собствени класове.
- 6.0 (1998 г.) – възможност за web програми.
- Net (2001 г.) – промяна в технологията.

ВБ_1

12 / 20

СЪБИТИЙНО ПРОГРАМИРАНЕ И ГПИ

Програмите, управлявани от събития, и системите за създаването им, **стават** твърде **популярни** след появата на ОС с ГПИ.

Причината се крие в **особеностите на ГПИ**:

- ❶ елегантно **решение на проблема** за създаване на **многозадачна ОС с един потребител**;
- ❷ **по-удобен за потребителите**, поради своята **интуитивна яснота**;
- ❸ **по-удобен и за производителите**, поради налагането на удобни **стандарты за общуване**.

ВБ_1

13 / 20

ЗАЩО ВБ?

Поради залагането на интуитивни принципи, **не е възможно** да има **няколко вида ГПИ**.

Така **системите за създаване на събитийни програми за ГПИ** могат да **се различават** само **по своя работен език и осигуряваните** наготово **класове от обекти**.

Основното **предимство на ВБ** се състои в това, че популярните **ОС с ГПИ – Windows**, и **системата за проектиране** са с **един и същ автор** – фирма **Майкрософт**, което означава **по-малък брой конфликти ОС – програма**.

ВБ_1

14 / 20

СТРУКТУРА НА ПРОГРАМАТА ПРИ ВБ

Програмата при ВБ е съвкупност от модули
от следните категории:

- ❶ **Форми**, реализиращи прозорците на ГПИ:
 - ➔ произволен брой **обикновени** (**Form**);
 - ➔ до една **за многодокументен интерфейс** (**MDI Form**);
 - ➔ **дъщерни на формата за МДИ** (**MDI Child**).
- ❷ **Стандартни** модули с обичайни ППГ.
- ❸ Модули за дефиниране на **вътрешни класове**.
- ❹ До един **ресурсен** модул.

ВБ_1

15 / 20

МОДУЛ ФОРМА

Всеки модул форма определя **специфичен вътрешен клас** от обекти и реализира ГПИ. Формите се въвеждат в ОП и се изобразяват **в отделни прозорци** в екрана (**обикновена** и **МДИ**) или в МДИ формата (**дъщерни**).

Носят върху себе си **елементите на ГПИ**.

Кодовата част на формата съдържа **общ** за нея **дефиниции** на променливи и обичайни процедури **и всички събитийни процедури** на разположените върху нея елементи.

ВБ_1

16 / 20

СТАНДАРТЕН МОДУЛ

Стандартните модули нямат ГПИ и съдържат **само програмен код**.

Записаните в стандартен модул **дефиниции** на именовани константи, променливи и процедури (функции) **са общи за цялата програма (проект)**, освен когато в записа явно не е посочено друго.

Както показва името на тези модули, **те са обичаен елемент** на модулното процедурно програмиране **и нямат връзка с ГПИ**.

ВБ_1

17 / 20

СТАРТИРАНЕ

Изпълнението на програмата може да започне **по** следните **два начина**:

- ❶ Въвеждане в ОП и показване на избрана **стартова форма**, т. е. **първи се** изпълнява **кодът на** нейното **събитие Initialize**;
- ❷ Изпълнение на **процедура Main** без видим графичен интерфейс с потребителя.

Показването на **допълнителни форми** се реализира чрез прилагане на **метод Show** към дадена форма, а въвеждането на форма в ОП – чрез **оператор Load** и при **достъп до свойство** на формата или носен от нея елемент на ГПИ.

ВБ_1

18 / 20

КРАЙ НА ПРОГРАМАТА

Програмата завършва изпълнение:

- ❶ при изпълнение на **оператори End и Stop.**
- ❷ при **извеждане от ОП на всички форми** и завършване на започналите процедури.

След оператор **End** не се изпълнява никакъв друг програмен код и **за наличните** в ОП **форми не възникват събитията**, свързани с извеждането им от ОП.

Извеждането на една форма от ОП става с изпълнение на **оператор Unload.**

ВБ_1

19 / 20

**БЛАГОДАРЯ ВИ
ЗА ВНИМАНИЕТО!**

**БЪДЕТЕ С МЕН И
В СЛЕДВАЩАТА ЛЕКЦИЯ,
КОЯТО ЩЕ НИ ОТВЕДЕ
В НЕВЕРОЯТНИЯ СВЯТ НА
ЕЗИКА ВБ, ВЕРСИЯ 6**