

Университет машиностроения
Кафедра «Автоматика и процессы управления»

Дисциплина
Информационные технологии
2 семестр

Тема 03

**Общие вопросы
обработки данных**
(Язык VBA for Ms Excel)

Элементы языка VBA

Особенности организации кода

Общие принципы

Visual Basic for Application – императивный язык программирования высокого уровня.

- Линейное:** Инструкции разделены и следуют последовательно.
- Структурное:** Блоки инструкций не содержат пересечений линий исполнения и собираются по принципу вложенности.
- Процедурное:** Часто повторяющиеся блоки инструкций могут быть выделены в подпрограммы (процедуры и функции) с одним входом и одним выходом.
- Модульное:** Часть кода приложения может быть вынесена в модули (библиотеки), подключаемые при необходимости использования.

Общие принципы

Visual Basic for Application – интерпретируемый язык программирования высокого уровня.

Компиляция: Построчный разбор кода при выполнении

Проверка: Синтаксис проверяется при вводе

Логические (структурные) ошибки выявляются только на этапе выполнения

Контроль работы с памятью

Хранение: Память может выделяться «по запросу» для использования с необъявленными именами

Типизация: Возможность изменения типа данных при исполнении

Необратимость: Изменения в данных невозможно обратить (вернуть)

Visual Basic for Application – язык с поддержкой объектно-ориентированного и событийного программирования

- Объект:** Совокупность **данных**, характеризующих его состояние, и **функций** их обработки, моделирующих поведение объекта.
- Класс:** Программный шаблон, на основе которого создается объект (реализация).
- Метод:** Функция или процедура, являющаяся частью описания объекта, предназначенная для выполнения каких-либо действий над объектом (данными).
- Событие:** Информационный эквивалент реакции системы на полученное сообщение.
- Обработчик:** Метод объекта, предназначенный для обработки специфического события.

Основной принцип организации кода в **Visual Basic for Application** – модульно-процедурный

Размещение кода: Модули в файлах .bas или в составе документа.

Организация кода: Исполняемые инструкции (команды) размещаются в теле процедур (Sub ... End Sub) или функций (Function ... End Function)

Объявление: В области General и внутри процедур и функций

Видимость: Доступность объекта или процедуры (функции) определяется размещением (например, в том же модуле, в той же процедуре и т.п.)

Модуль ≈ класс объекта (ООП)

Private, public управляют доступом к свойствам (глобальным переменным модуля) и методам (процедурам и функциям)

Элементы языка VBA

Операции с данными Хранение и преобразование

Представление данных в коде

Литералы

Литеральная константа – это данные (число, строковое выражение, дата и т.п.), размещенные непосредственно в коде программы

Константы

Именованная константа – это какие-либо данные, которые не изменяются при выполнении программы, и для обращения к ним используется специальное символьное имя определенное в коде

Допускается использование **типизированных констант** при объявлении которых явно задаётся тип данных

Переменные

Именованные объекты, предназначенные для временного хранения изменяемых данных

Специальные константы

Внутренняя константа – это именованная константа, которая была определена разработчиками VBA.

Внутренние константы для работы с host-приложениями.
Excel содержит внутренние константы для использования с рабочими книгами электронных таблиц.

Word содержит внутренние константы для работы с документами и шаблонами текстового редактора

Access – константы для операций с базами данных.

Внутренние константы, определяемые VBA, начинаются с букв **vb**.

Внутренние константы Excel - **xl**; Word - **wd**.

Полный список имеющихся в наличии внутренних констант доступен через **Object Browser**.

Его можно вызвать клавишей F2 в окне редактора VBA.

Использование литералов и констант

Начало и конец кода макроса

Наименование макроса (может быть записано кириллицей)

Комментарий к тексту программы

Sub Vvod_Formuly_Skidki()

' Макрос записан 1.12.2012

' Быстрый вызов Ctrl + Q

Объявление типизированной константы

Const A As Integer = 1

Строковый литерал – последовательность символов в коде программы, интерпретируемых как данные

ActiveCell.Value = "Hello world"

Числовой литерал

Cells(ActiveCell.Row, ActiveCell.Column + 1).FormulaR1C1 = "(RC[-2]*RC[-3]-RC[-1])*usd"

Cells(ActiveCell.Row + A, ActiveCell.Column).Select

End Sub

Использование константы

Метод

Объект

Свойство

Объявление переменных

Оператор **Dim** (от **dimention**) – Объявляет и размещает в памяти одну или несколько переменных.

Область
General
модуля

Dim i

Объявление нетипизированной
глобальной переменной

Dim j As Integer

Объявление типизированной
глобальной переменной

Sub MySub()

' Макрос записан 1.12.2012

' Быстрый вызов Ctrl + Q

Объявление типизированной
переменной и инициализация
значением

Dim X As Double = 0.5

Неявное объявление через
установку (присвоение)
значения

Y = 1 + X

Использование переменной

ActiveCell.Value = "1" + Y

Неявное преобразование типов

End Sub

Оператор присваивания

Типы данных

Тип данных	Резервируется байт	Наименьшее значение	Наибольшее значение
Byte	1	0	255
Boolean	2	False (Ложь)	True (Истина)
Integer	2	-32768	32767
Long	4	-2147483648	2147483647
Single	4	-3.402823 E38	-1.401298 E-45
		1.401298E-45	3.402823E38
Double	8	-1,79769313486232E308	-4,94065645341247E-324
		4,94065645841247E-324	1,79769313486232E308
Currency	8	-922337203685477,5808	922337203685477,5807
Decimal	14	+/-79228162514264337593543950335 без десятичных знаков	+/-7,9228162514264337593543950335 с 28-ью знаками после запятой
Date	8	1 января 100 года	31 декабря 9999 года
Object	4	Любая ссылка на объект	
String (переменной длины)	10 байт + длина строки	0	приблизительно 2 млрд
String (фиксированной длины)	Длина строки	1	65400
Variant (числа)	16	Любое числовое значение в рамках диапазона типа данных Double	
Variant (символы)	22 байта + длина строки	0	приблизительно 2 млрд
Пользовательский	Зависит от типа	Зависит от элемента	

«Наследие Basic» - явное указание типа

Специальный символ (из не разрешенных к использованию в именах переменных) выступает **указателем типа** переменной при объявлении и использовании.

Dim X# ← Использование указателя типа переменной

Dim Y As Double ← Объявление типизированной переменной

Символы объявления типов:

Тип данных	Символ объявления типа
Integer	%
Long	&
Single	!
Double	#
Currency	@
String	S

Операции с переменными

Применение модификаторов области видимости и способа размещения в памяти

- public** – переменная доступна во всех модулях приложения
- private** – переменная доступна только в данном модуле (аналогично **Dim** в **General**)
- static** – переменная размещается в статической памяти



Хранение данных:

Как это обычно работает?

- **Регистры процессора**
- **Структура памяти приложения**
- **Стек, куча и указатели**



Сегментные регистры

- CS (Code Segment)
- DS (Data Segment)
- SS (Stack Segment)
- ES (Enhanced Segment)

Регистры-указатели

- SI (Source Index)
- DI (Destination Index)

Регистры-указатели

- BP (Base Pointer)
- SP (Stack Pointer)

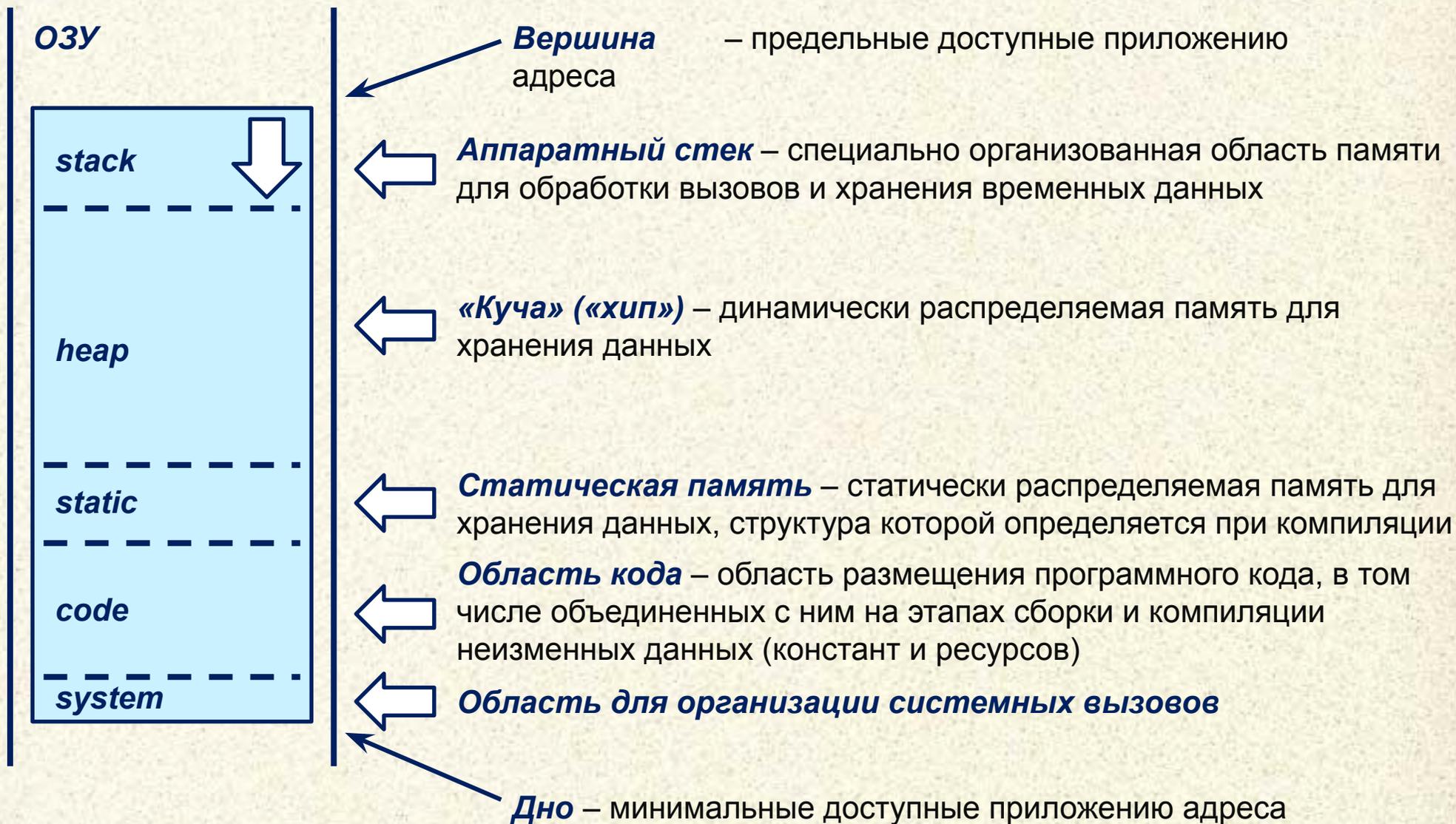
Assembler (сборщик) – язык низкого уровня, основанный на мнемонической записи инструкций центрального процессора

MOV A,100 – поместить число 100 в регистр аккумулятора

PUSH C – поместить (втолкнуть) регистр C в стек

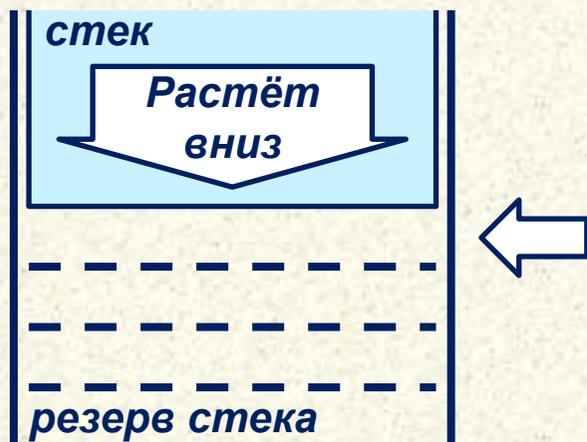
Структура памяти приложения

Размещение в памяти зависит от способа объявления переменной и типа данных



Использование стека при вычислениях

При использовании стека **необходимо следить** за используемыми размерами типов данных



В стековых языках программирования стек используется для размещения данных, а адреса ячеек рассчитываются относительно вершины стека

Вершина стека – указатель на первую свободную ячейку

Адрес вершины стека хранится в специальных регистрах процессора **SS** (селектор регистра стека) и **ESP** (указатель стека)

Инструкция **Push** (Втолкнуть) используется для внесения в стек промежуточных данных и ссылок, автоматически уменьшает **ESP**

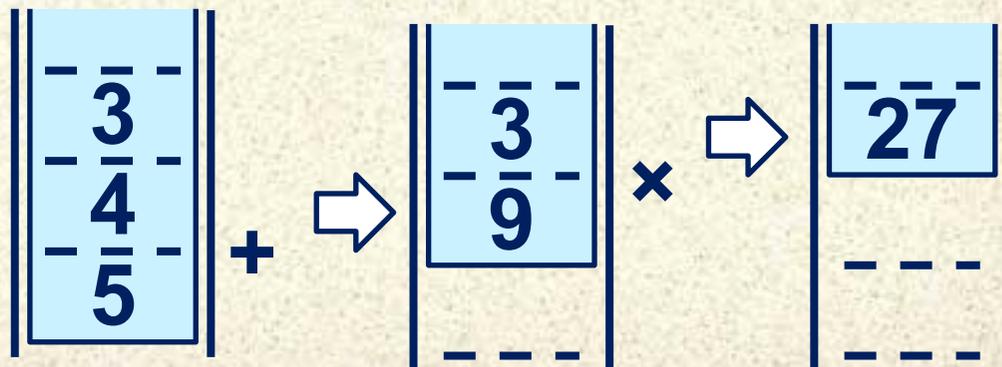
Инструкция **Pop** (Извлечь) используются для получения из стека промежуточных данных и ссылок, автоматически увеличивает **ESP**

Обратная польская нотация позволяет записать математическое выражение в виде последовательности данных (операндов) отдельных бинарных и унарных операций

$$3 \times (4 + 5)$$

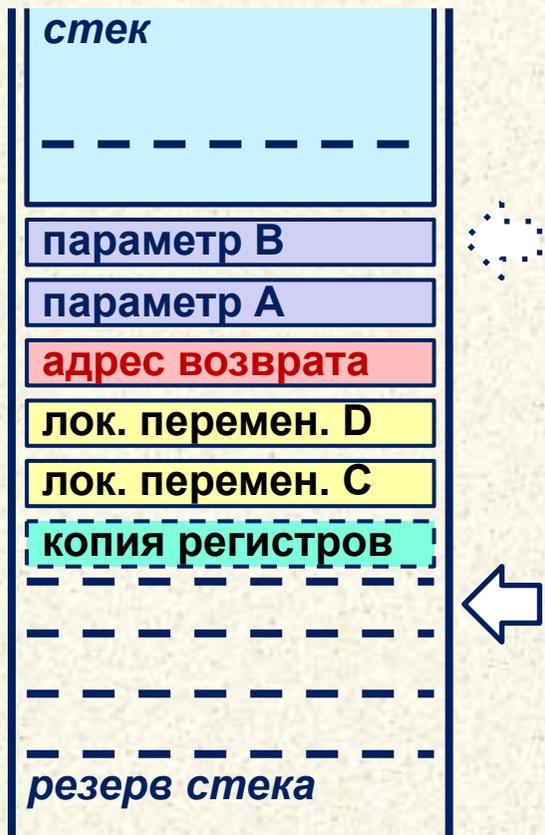


$$3 \ 4 \ 5 \ + \ \times$$



Поддержка стековых операций процессором

Базовое использование стека – сохранение состояния процессора (всех регистров) при **вызове подпрограммы** или **обработке прерывания**



Инструкция **Call** (Вызвать) помещает в стек **адрес выборки команды**, затем выполняет сохранение **всех регистров**, а потом выполняется **переход по адресу**, указанному в инструкции

Interrupt (Прерывание) – встроенная последовательность операций процессора, выполняемая в ответ на заданное событие. При обработке прерывания в стек помещается текущий **адрес выборки команды**, затем по номеру прерывания вычисляется ячейка в **таблице прерываний**, содержащая **адрес процедуры обработчика** и выполняется переход по этому адресу

Offset (Смещение) – возможность указать в инструкции по обмену данными адрес ячейки памяти с помощью задания смещения от **адреса вершины стека**. Например, **ADD AX, [SP + offset]**
С учетом возможности прямого изменения значения указателя стека это даёт возможность выделять (резервировать) память под хранение данных

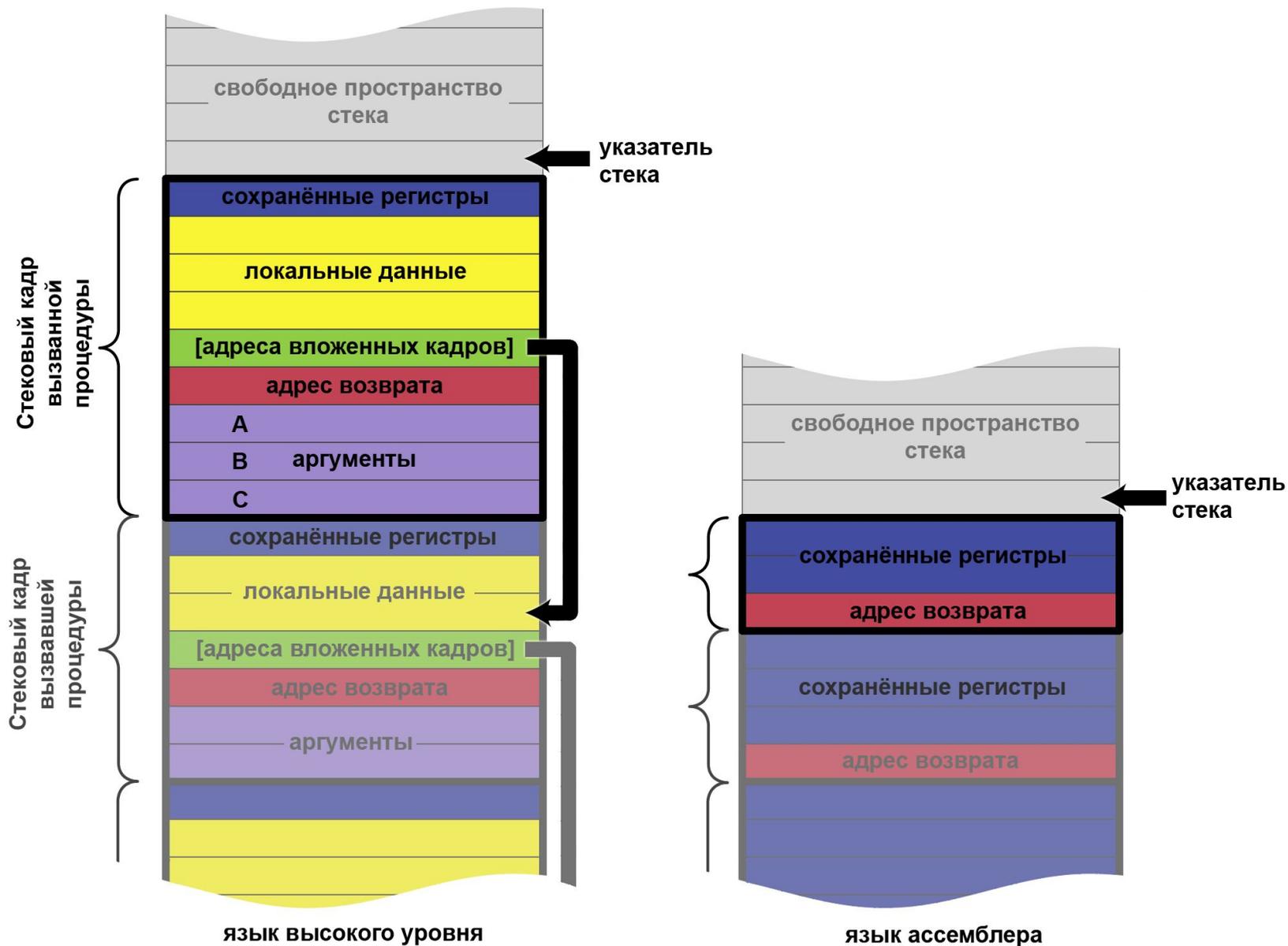
В стековых языках программирования стек используется для размещения данных, а адреса ячеек в коде рассчитываются относительно вершины стека

Для i8086 для сохранения копии регистров нужно 26 байт, поэтому (в подпрограмме):

MOV AX, [SP + 34] – поместить в аккумулятор (регистр AX) значение **параметра А**

Задача: запишите на ассемблере **D = A + B**

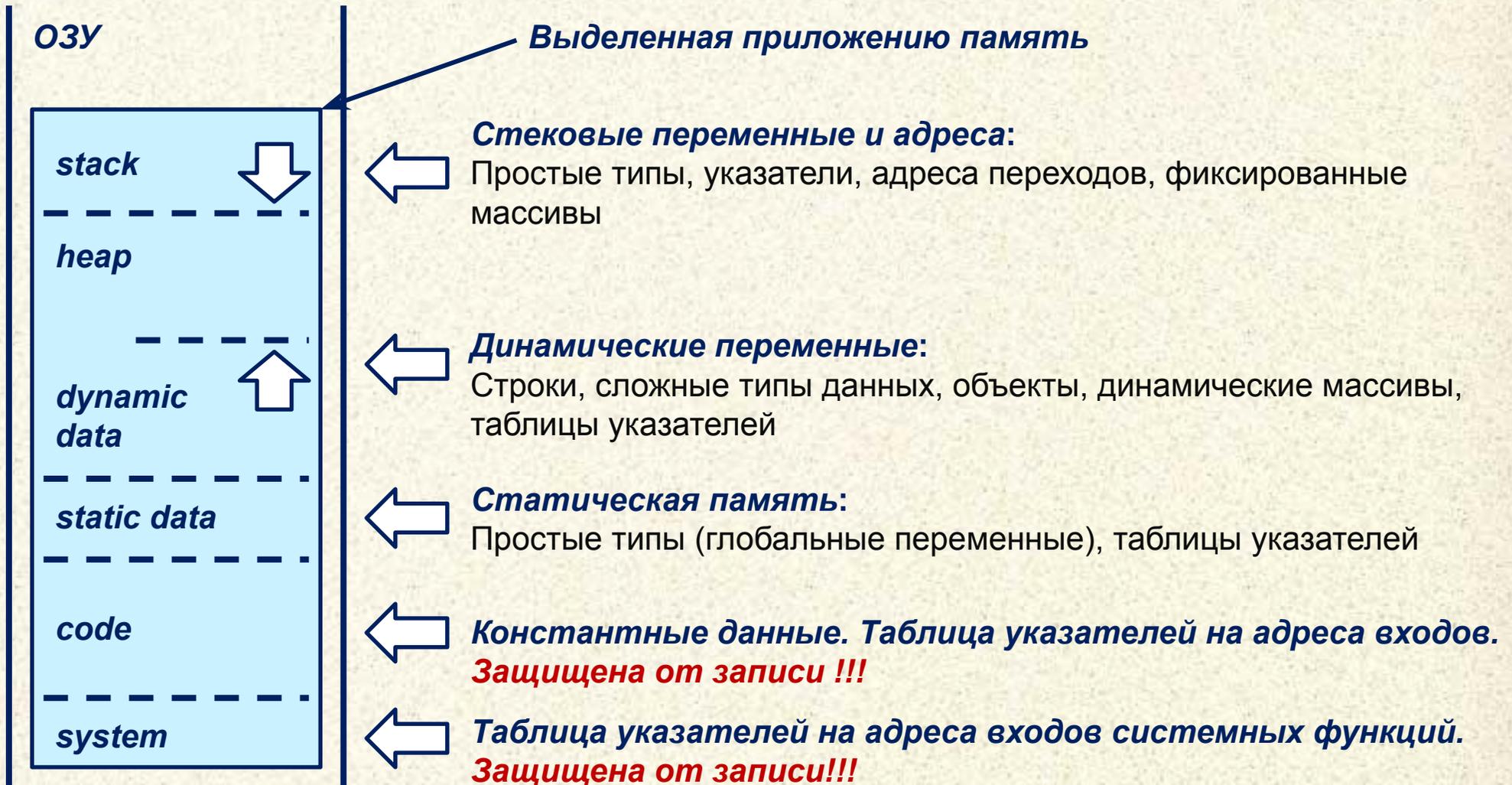
Практическое использование стека



P.S. Изображение перевернуто

Где хранятся переменные?

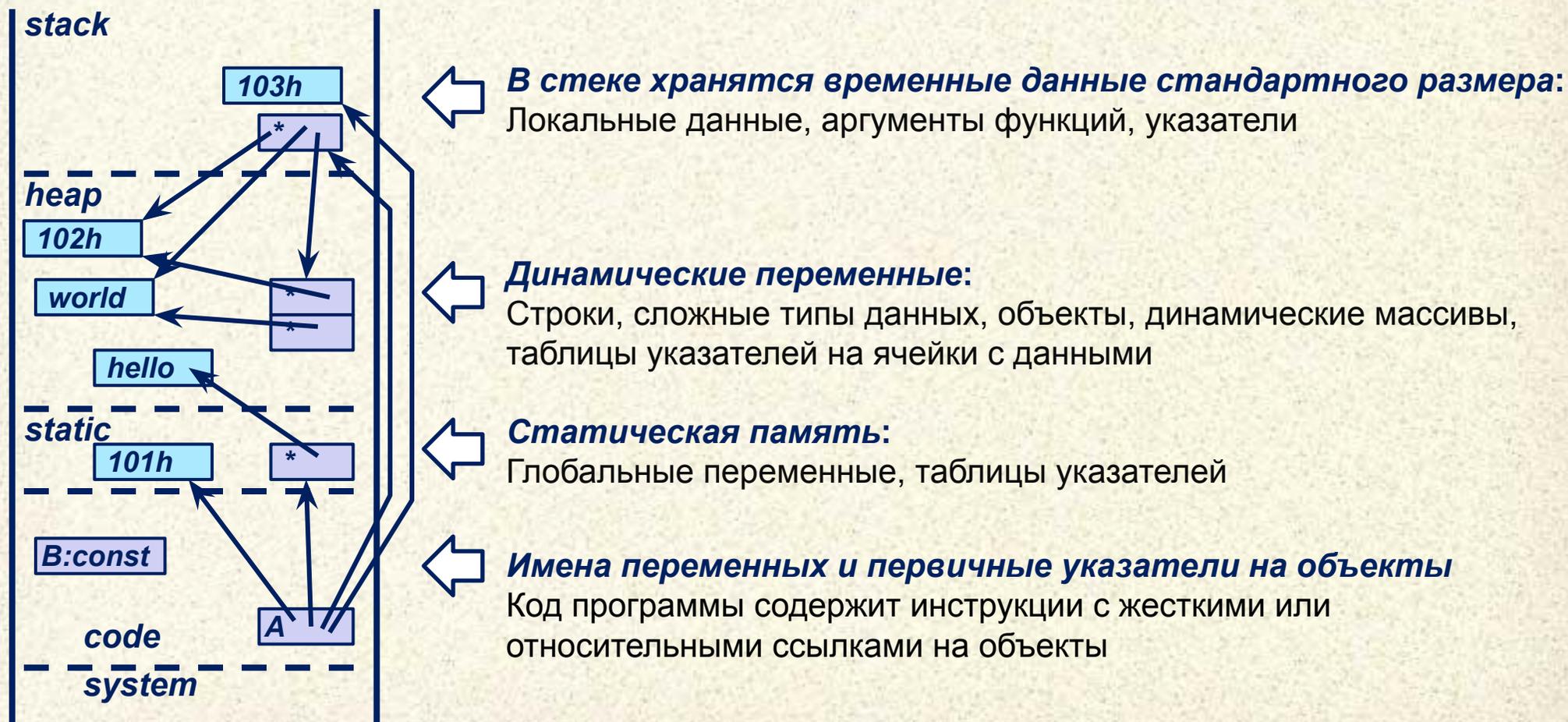
Размещение в памяти зависит от способа объявления переменной и типа данных



Как хранятся переменные?

Метод хранения зависит от структуры данных:

- Простые данные (integer, boolean, datetime...) могут храниться в одной ячейке;
- Составные данные (массивы) фиксированного размера могут храниться компактно;
- Данные сложной структуры хранятся с использованием таблицы указателей;
- Все динамические данные хранятся с использованием указателей.



Общие правила работы с переменными и объектами

Объявление

Присвоение символического (читаемого) имени структуре данных заданного типа.

Объявление простого типа в VBA эквивалентно выделению памяти и инициализации «пустым» (нулевым) значением

Инициализация

Физическое выделение памяти с одновременным вводом начального (или заданного) значения

Динамические структуры данных могут инициализироваться без выделение памяти под данные, а копированием указателя (принцип: копирование по запросу)

Изменение

Простые типы фиксированного размера перезаписывают значение в ту же ячейку памяти

Динамические и сложные типы заново выделяют память и копируют неизменившиеся фрагменты

Освобождение

«Сборщик мусора» для «ненужных» ячеек в куче (не используемых далее в коде или разыменованных явно)

Особенности (отличия) VBA

Интерпретация

Построчная обработка исходного кода программы с возможностью приостановки исполнения (паузы) и переключения в режим «отладки»

Код макросов исполняется в одном потоке с функциями хост-приложения и пользовательских элементов управления, при этом отладочные инструменты работают в отдельном потоке

Отладка

Средства отладки доступны всё время исполнения кода программы и в режиме паузы, в котором возможны модификация исходного кода, а также значений переменных

«Постановка на паузу» возможна из меню, либо с использованием «точек останова» (breakpoints)

Автоматическое освобождение памяти – все ссылки на объекты автоматически разыменовываются при завершении исполнения блока кода, в котором были объявлены данные переменные-указатели

Созданные объекты уничтожаются автоматически

Спасибо за внимание !!!

Далее:

- Введение в ПиОА
- Структурные диаграммы
- Общие вопросы проектирования
- Обработка данных
- Основные алгоритмические конструкции

...