Оглавление

[Лекции по информатике – 1 семестр ВКБ11 2](#_Toc121695554)

[Лекция 1 - Цифровые технологии в образовательном процессе. Сквозные технологии в современном мире. 2](#_Toc121695555)

[Лекция 2 - Теоретические основы информатики 3](#_Toc121695558)

[Лекция 3 - Системы счисления 5](#_Toc121695559)

[Лекция 4 - Логические основы компьютера 6](#_Toc121695560)

[Лекция 5 - Понятие архитектуры и структуры компьютера. Организационная структура микропроцессора. 6](#_Toc121695561)

[Лекция 6 - Операционные системы. Введение. 8](#_Toc121695562)

[Лекция 7 - Управление памятью 11](#_Toc121695563)

[Лекция 8 - Основные понятия баз данных. 20](#_Toc121695564)

[Лекция 9 - Основы информационной безопасности 25](#_Toc121695565)

[Лекция 10 - Основы программирования в Python 30](#_Toc121695566)

[Лекция 11 - Вывод и ввод информации в среде программирования Python. 34](#_Toc121695571)

[Лекция 12 - Разработка графического интерфейса в среде программирования Python 37](#_Toc121695575)

[Лекция 13 - Циклические вычислительные конструкцию. Обработка исключений. Текстовые строки в Python 41](#_Toc121695576)

# Лекции по информатике – 1 семестр ВКБ11

## Лекция 1 - Цифровые технологии в образовательном процессе. Сквозные технологии в современном мире.

**Технологии –** один из драйверов изменений;

Технологии – применение научного знания для решения практических задач; это искусство, умение или способность, которые используются для создания и развития продуктов, приобретение знаний.

### Цикл «Хайпа»

*Любая технология проходит этот цикл*

1. **Технологический триггер *–*** появление инновации, первые публикации о новой технологии (низшая точка на графике)
2. **Пик чрезмерных ожиданий** – от технологии ожидают революционных свойств (высшая точка на графике)
3. **Избавление от иллюзий** – выявление недостатков технологии, наступает разочарование (спад на графике)
4. **Преодоление недостатков** – технология начинает внедряться в коммерческие проекты (возрастание на графике)
5. **Плато продуктивности** – зрелость технологии, она воспринимается как данность

### Выделяют сквозные технологии:

1. Нейротехнологии и ИИ
2. Распределённые реестры
3. Квантовые технологии
4. Технологии «Больших данных»
5. Интернет вещей
6. Беспроводная связь
7. VR и AR
8. Робототехника и сенсорика
9. Облачные технологии (облака)
10. Новые производственные технологии (digital twin, manufacturing)

**Данные** – это переработанная информация

|  |  |  |
| --- | --- | --- |
| Собирать данные | Передавать и хранить данные | Анализ и принятие решений |
| IoT | 5G | ИИ |
| Big Data | Квантовые технологии | Нейротехнологии |
|  | Блокчейн |  |
|  | Облака |  |

**Интернет вещей** – это датчики, сенсоры, различные носимые устройства, подключённые к интернету (# Часы, браслеты, смартфоны…)

**Big Data** – это разнообразные данные, которые поступают с растущей скоростью и объём которых постоянно растёт. (#Персонализация предложений для клиентов+

**Облачные технологии** – это хранение, обработка больших объёмов данных, доступ к онлайн сервисам 24/7; тесная взаимосвязь с интернет вещами.

**Искусственный интеллект** – это программы и устройства, имитирующие интеллектуальные функции человека (анализ данных, принятие решений); обучение на основе размеченных человеком данных. **Блокчейн** – это выстроенный в хронологической последовательности список взаимосвязанных записей, которые собираются в цепочки (информация устойчива, в классификации не требуется посредников).

## Лекция 2 - Теоретические основы информатики

**Информатика** – это наука о вычислении, хранении и обработки информации

**Информационное общество** – это историческая фаза развития цивилизации, в которой главными продуктами производства являются информация и знания

**Отличительными чертами являются:**

1. Увеличение роли информации и знаний
2. Возрастание доли информационных коммуникаций, продуктов и услуг в ВПП
3. Создание глобального информационного пространства, **обеспечивающего:**
   1. Эффективное информационное взаимодействие
   2. Доступ к мировым информационным ресурсам
   3. Удовлетворение потребностей в информационных продуктах и услугах

Процесс проникновения информационных технологий во все сферы жизни называется **процессом информатизации общества**. Информатизация общества является современным социальным прогрессом.

Следует различать понятия:

1. **Компьютеризация**

При компьютеризации объекта или общества основное внимание уделяется развитию и внедрению технической базы в виде компьютеров для оперативной обработки информации.

1. **Информатизация**

При информатизации акцент делается на комплекс мер по обеспечению доступа каждого человека к накопленному и имеющемуся в системе информационному ресурсу. Под **информатизацией** можно понимать организованный социально-экономический и научно-технический процесс создания оптимальных условий для удовлетворения информационных потребностей.

**Информация** – это сведения, разъяснения, изложения

**Под информацией в технике** принято понимать любую последовательность символов, знаков, сигналов, не учитывая их смысл.

**Для человека информация** – это знания, ощущения, накопленный опыт

**Информационные технологии** – это процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов

**Информационная система** – это совокупность содержащейся в базах данных информации и обеспечивающих её обработку информационных технологий и технологических средств

**Информатика классифицируется:**

1. По способам восприятия (визуальная)
2. По способам представления (текстовая, …)

**Информационные процессы** – это процессы, связанные с поиском, передачей, обработкой, хранением и использованием информации

**Поиск информации** – это извлечение хранимой информации

**Методы поиска информации:**

1. Ручной
2. Автоматизированный

**Обработка информации** – это процесс преобразования информации из одного вида в другой

**Обработка информации может происходить:**

1. По строгим формальным правилам
2. По принципу чёрного ящика

**Хранение информации** – это способ распространения информации в пространстве и времени

**Носитель информации** – это среда для записи и хранения информации

**Информация может существовать:**

1. В образной форме
2. В символьной // знаковой форме

**Язык** – это система знаков и правил их использования для представления информации

**Естественные языки** – это разговорные, возникшие и развивающиеся вместе с носителем языка

**Искусственные** // **Формальные языки** – это специально созданные для представления любого вида информации в определённой области деятельности

**Компьютер** – универсальный инструмент для работы с информацией, который использует двоичный код

Хранение и обработка информации реализованы в двоичных кодах с применением двоичной системы счисления. Это связано с многоразрядностью электронных схем памяти, каждый разряд которых – бит. Может принимать всего два состояния: «0» или «1». Объём информации, записанной двоичными знаками в памяти компьютера, подсчитывается по числу требуемых для такой записи двоичных символов.

В компьютере, бит – наименьшая возможная единица информации. Для удобства введены более крупные единицы (Петабайт…)

Между **вероятностным и объёмным количеством** информации соотношение неоднозначное. В прикладной информатике количество информации понимается в объёмном смысле.

## Лекция 3 - Системы счисления

**Системы счисления** – это способ представления любого числа с помощью некоторого алфавита символов, называемых цифрами.

Системы счисления бывают: **Позиционные** и **Непозиционные**

Чтобы записать число в разных системах счисления нужно использовать некоторое количество отличных друг от друга знаков.

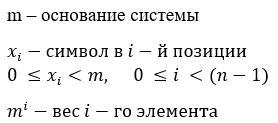
**В позиционной системе счисления** число таких знаков называется основанием системы счисления

|  |  |  |
| --- | --- | --- |
| Основание | Система счисления | Знаки |
| 2 | Двоичная | 0 1 |
| 3 | Троичная | 0 1 2 |
| 4 | Четверичная | 0 1 2 3 |
| 5 | Пятеричная | 0 1 2 3 4 |
| 6 | Шестеричная | 0 1 2 3 4 5 |
| 16 | Шестнадцатеричная | 0 1 2 3 4 5 6 7 8 9 A B C D E F |

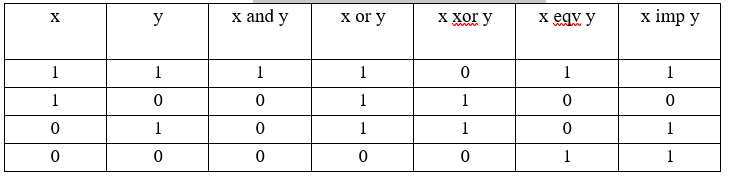
**В позиционной системе счисления** значимость (вес) каждой цифры числа зависит от позиции, которую она занимает

Значение числа, состоящего из n цифр может быть определено следующим образом:





## Лекция 4 - Логические основы компьютера



## Лекция 5 - Понятие архитектуры и структуры компьютера. Организационная структура микропроцессора.

Логически системную шину можно разделить на **три шины**:

1. **Шина данных** – по ней идёт информация (команда и данные)
2. **Адресная шина** – по ней идут коды с использованием, которых находится нужная ячейка в ОЗУ или нужный порт ввода-вывода
3. **Шина управления** – через которую передаются управляющие сигналы типа “что делать”, “запись в память”.

**Протокол работы** системной шины состоит из **четырёх пунктов**:

1. **Первый такт** – процессор **выставляет** на шину адреса **адрес** ячейки памяти или порта внешнего устройства и устанавливает на шине управления сигналы, определяющие тип обмена.
2. **Второй такт** – процессор получает **сигнал** **готовности** выбранного устройства к приёму или передачи информации, если сигнал готовности не получен второй такт может повторяться бесконечно.
3. На **третьем такте** процессор или **открывает шину для приёма данных**, или при записи выставляет на шину данных передаваемую информацию.
4. На **четвёртом такте** происходит **обмен информацией**, и работа протокола передачи заканчивается.

Для того чтобы различные производители могли выпускать совместимое оборудование обязательно должны соблюдаться соглашения о следующих параметрах:

1. Геометрические размеры слота, число и расположение контактов.
2. Материнская плата и системная плата работают в цифровом режиме, они обмениваются битами информации. **Физический сигнал, соответствующий биту информации**, представляет собой импульс напряжения определённой величины.
3. Далее надо договориться какая последовательность битов, т.е. код, соответствует какой команде, чтобы принимающая микросхема “знала”, что ей делать

**Интерфейс** – соглашение о механических, физических и логических параметров слота.

Стандарты, получившие наибольшее распространение:

1. **PCI** (стандарт подключения внешних компонентов) был введён на базе процессоров Intel Pentium. Это интерфейс локальной шины, связывающий процессор с оперативной памятью, в которую врезаны разъёмы для подключения внешних устройств
2. **PCMCIA** применяется в переносных пк класса ноут и имеет параметры сравнимые с параметрами шины PCI.
3. **AGP** локальная шина, введённая для повышения производительности графической подсистемы компьютера. Позволяет организовать связь видеоконтроллера и оперативно запоминающего устройства
4. **PCI Express** является пакетной сетью с топологией звезда. Использует для передачи данных общую шину. Устройства (эти) взаимодействуют между собой через среду, образованную коммутаторами.

Разные модели процессоров могут использовать разные шины, поэтому необходимы устройства для сопряжения шины процессора с шиной периферийных устройств (**контроллеры, мосты**). Данные адаптеры включают специализированные наборы системной логики, называемые северный мост (котроллер оперативной памяти или системный контроллер) и южный мост (контроллер периферийных устройств). **Северный мост обеспечивает обмен операцией между процессором и оперативной памятью** по системной шине. Набор из южного и северного мостов называют **чипсетом**.

**К северному мосту подключается шина PCI** (шина взаимодействия периферийных устройств), которая обеспечивает обмен информации с контроллерами периферийных устройств. **Частота контроллеров меньше частоты системной шины**, например если частота шины составляет 100 МГц, то частота шины PCI обычно в 3 раза меньше – 33МГц. Контроллеры периферийных устройств устанавливается в слоты расширения системной платы.  
По мере увеличения разрешающей способности монитора и глубины цвета требования к быстродействию шины, связывающей видеоплату с процессором и оперативной, возрастают.

**Южный мост обеспечивает обмен информации между северным мостом и портами для подключения периферийного оборудования**.

Устройства хранения информации подключаются к южному мосту по шине **UDMA**.

Мышь и внешний модем подключаются к южному мосту с помощью последовательных портов, которые передают электрические импульсы, несущие некоторую информацию в машинном коде, последовательно один за другим.

Принтер подключается к параллельному порту, который обеспечивает высокую скорость передачи, чем последовательные порты так как передаёт одновременно 8 электрических импульсов.

Процессор – устройство, предназначенное для выполнения команд.

Различают универсальные и специализированные.

**Специализированные**:

1. Проблемно-ориентированные
2. Функционально-ориентированные
3. Системные

**Микропроцессор выполняет функции:**

* Выполняет арифметические и логические операции
* Управляет вычислительными процессами
* Координирует работу всех устройств компьютера

**Центральный процессор содержит:**

1. Арифметико-логическое устройство
2. Устройство управления
3. Шины данных и шины адресов (на физическом уровне) – много проводные линии с гнёздами для подключения электронных схем.
4. Регистры – ячейки памяти, которые служат для кратковременного хранения и преобразования данных и команд (на физическом уровне совокупность триггеров, способных хранить один двоичный разряд и связанных между собой общей системой управления).
5. Счётчик команд – регистр управляющего устройства компьютера, содержимое которого соответствует адресу очередной выполняемой команды. Счётчик команд служит для автоматической выборки программы из последовательных ячеек памяти.
6. Кэш-память – очень быстрая память малого объёма. Она служит для увеличения производительности компьютера путём согласования устройств, работающих на различной скорости.
7. Сопроцессор – вспомогательный процессор, предназначенный для выполнения математических и логических действий.
8. Интерфейсная часть микропроцессора – реализует сопряжение и связь с другими устройствами пк. Он обменивается информацией с другими устройствами через поры ввода-вывода Всего портов у МП - . Каждый порт имеет адрес – номер порта, соответствующий адресу ячейки памяти.

**Основными характеристиками микропроцессора являются**:

* Тактовая частота
* Разрядность
* Архитектура

## Лекция 6 - Операционные системы. Введение.

**Операционная система** (operating system)– комплекс управляющих и обрабатывающих программ, которые, с одной стороны, выступают как интерфейс между устройствами вычислительной системы и прикладными программами, а с другой – предназначены для управления устройствами, управления вычислительными процессами, эффективного распределения вычислительных ресурсов между вычислительными процессами и организации надёжных вычислений.

**Современное универсальные ОС можно охарактеризовать по следующим признакам:**  
1) Использующие файловые системы (с универсальным механизмом доступом к данным)

2) Многопользовательские (с разделением полномочий)

3) Многозадачные (с разделением времени)

Многозадачность и распределение полномочий требуют определённой иерархии привилегий компонентов в самой операционной системе

**Unix, стандартизация операционных систем**

Unix – распространённая ОС 90-ых годов, бесплатно, легко модифицируемо

**Классификация операционных систем:**

1. По базовой технологии (Unix подобные, потомки Unix)
2. По типу лицензии (проприетарная или открытая)
3. Устаревшие или современные
4. По назначению (универсальные, ОС встроенных систем, ОС PDA, OC реального времени, для рабочих станций или для серверов) и др

**Операционные системы нужны:**

1. Если нужен универсальный механизм **сохранения данных**
2. Для **предоставления** системным **библиотекам** часто использующих **подпрограмм**
3. Для распределения **полномочий**
4. Если необходима возможность **имитации “одновременного “ исполнения** нескольких программ на одном компьютере
5. Для **управления процессами** выполнения отдельных программ

**В составе ОС различают три группы компонентов:**

1. Ядро, содержащее планировщик; драйверы устройств, управляющие оборудованием; сетевая подсистема, файловая система
2. Системные библиотеки
3. Оболочка с утилитами

**Операционные функции операционных систем:**

* Выполнение по запросу программ
* Загрузка программ в оперативную память и их выполнение
* Стандартизированный доступ к периферийным устройствам
* Управление оперативной памятью
* Управление доступом к данным на энергонезависимых носителях
* Обеспечение пользовательского интерфейса
* Сохранение информации об ошибках системы

**Дополнительные функции операционных систем:**

* Параллельное или псевдопараллельное выполнение задач
* Эффективное распределение ресурсов вычислительной системы между процессами
* Разграничение доступа различных процессов к ресурсам
* Организация надёжных вычислений основана на разграничении доступа к ресурсам
* Взаимодействие между процессами: обмен данными, взаимная синхронизация
* Защита самой системы, а также пользовательских данных и программ от действий пользователей или приложений
* Многопользовательский режим работы и разграничение прав доступа

**Компоненты операционной системы:**

* Загрузчик
* Ядро
* Командный процессор (интерпретатор)
* BIOS
* Драйверы устройств
* Интерфейс

**Ядро операционной системы** – это центральная часть, управляющая выполнением процессов, ресурсами вычислительной системы и предоставляющая процессам координированный доступ к этим ресурсам

**Объекты ядра операционной системы:**

* Процессы
* Файлы
* События
* Потоки
* Семафоры
* Мьютексы
* Каналы
* Файлы, проецируемые в память

**Ядро операционной системы** –основными ресурсами являются **процессорное время, память и устройства ввода-вывода**. Доступ к файловой системе и сетевое взаимодействие могут быть реализованы на уровне ядра.

Ядро представляет собой наиболее низкий уровень абстракции для доступа приложений к ресурсам вычислительной системе.

Ядро предоставляет такой доступ исполняемым процессам соответствующих приложений за счёт использования механизмов межпроцессного взаимодействия и обращения приложений к системным вызовам ОС.

**Оболочка (интерпретатор) командной строки command.com || cmd.exe**

Во всех версиях ОС Windows поддерживается интерактивная оболочка командной строки (command shell) и определённый набор утилит командной строки.

Механизм работы оболочек командной строки в разных системах одинаков. В ответ на приглашение (prompt), выдаваемое находящийся в ожидании оболочкой, пользователь вводит некоторую команду, оболочка выполняет её при необходимости выводя на экран какую-либо информацию, после чего снова выводит приглашения и ожидает ввода следующей команды.

Оболочка представляет собой построчный интерпретатор простого языка (директивного) программирования, в качестве операторов которого могут использоваться исполняемые программы.

Наряду с интерактивным режимом работы оболочки, как правило, поддерживается и пакетный режим, в котором система последовательно выполняет команды, записанные в текстовом файле-сценарии.

Оболочка командной строки представляется интерпретатором cmd.exe

**Новая оболочка PowerShell**

Основана на идее, что в командной строке вывод результата представляет собой не текст, а объект (не последовательность символов, а данные вместе со свойственными им методами)

## Лекция 7 - Управление памятью

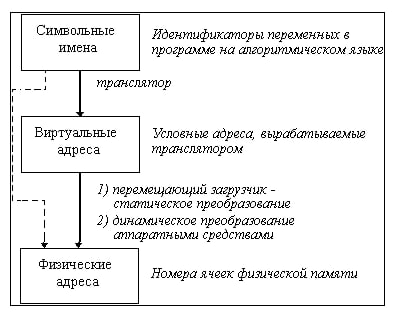
1. **Типы адресов**

Память является важнейшим ресурсом, требующим тщательного управления со стороны мультипрограммной операционной системы. Распределению подлежит вся оперативная память, не занятая операционной системой.

*Функциями операционной системы* по управлению памятью являются:

* Отслеживание свободной и занятой памяти
* Выделение памяти процессам и освобождение памяти при завершении процессов
* Вытеснение процессов из оперативной памяти на диск, когда размеры основной памяти недостаточны для размещения в ней всех процессов
* Возращение их в оперативную память, когда в ней освобождается место
* (а также) Настройка адресов программы на конкретную область физической памяти

Для идентификации переменных и команд используются символьные имена (метки), виртуальные адреса и физические адреса (рисунок 1)



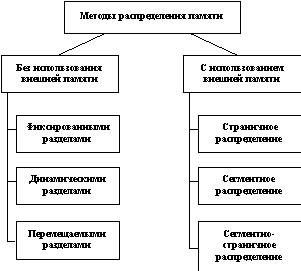
Символьные имена присваивает пользователь при написании программы на алгоритмическом языке или ассемблере.

Виртуальные адреса вырабатывает транслятор, переводящий программу на машинный язык.

Физические адреса соответствуют номерам ячеек оперативной памяти, где в действительности расположены или будут расположены переменные и команды.

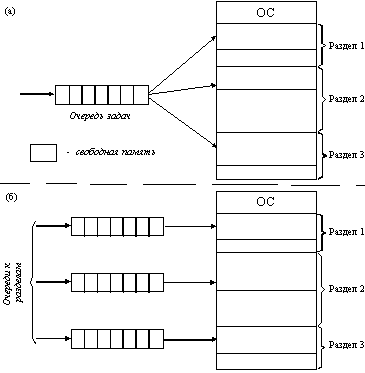
1. **Методы распределения памяти без использования дискового пространства**

Все методы управления памятью могут быть разделены на два класса: методы, которые используют перемещение процессов между оперативной памятью и диском, и методы, которые не делают этого (рисунок 2 – Классификация методов распределения памяти)



1. **Распределение памяти фиксированными разделами**

Самым простым способом управления оперативной памятью является разделение ее на несколько разделов фиксированной величины. Это может быть выполнено вручную оператором во время старта системы или во время ее генерации. Очередная задача, поступившая на выполнение, помещается либо в общую очередь (рисунок 3, а), либо в очередь к некоторому разделу (рисунок 3, б).



Подсистема управления памятью в этом случае выполняет следующие задачи:

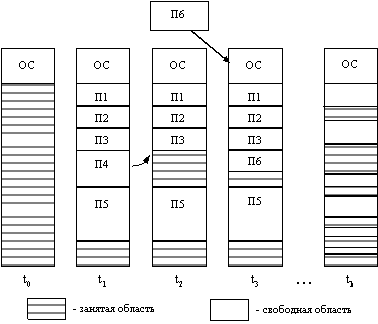
* сравнивая размер программы, поступившей на выполнение, и свободных разделов, выбирает подходящий раздел,
* осуществляет загрузку программы и настройку адресов.

При очевидном преимуществе - простоте реализации - данный метод имеет существенный недостаток - жёсткость.

1. **Распределение памяти разделами переменной величины**

В этом случае память машины не делится заранее на разделы. Сначала вся память свободна. Каждой вновь поступающей задаче выделяется необходимая ей память. Если достаточный объем памяти отсутствует, то задача не принимается на выполнение и стоит в очереди. После завершения задачи память освобождается, и на это место может быть загружена другая задача.

На рисунке 4 показано состояние памяти в различные моменты времени при использовании динамического распределения. Так в момент t0 в памяти находится только ОС, а к моменту t1 память разделена между 5 задачами, причем задача П4, завершаясь, покидает память. На освободившееся после задачи П4 место загружается задача П6, поступившая в момент t3.



Задачами операционной системы при реализации данного метода управления памятью является:

* Ведение таблиц свободных и занятых областей, в которых указываются начальные адреса и размеры участков памяти,
* При поступлении новой задачи - анализ запроса, просмотр таблицы свободных областей и выбор раздела, размер которого достаточен для размещения поступившей задачи,
* Загрузка задачи в выделенный ей раздел и корректировка таблиц свободных и занятых областей,
* После завершения задачи корректировка таблиц свободных и занятых областей.

По сравнению с методом распределения памяти фиксированными разделами данный метод обладает гораздо большей гибкостью, но ему присущ очень серьезный недостаток - *фрагментация памяти*. Фрагментация — это наличие большого числа несмежных участков свободной памяти очень маленького размера (фрагментов). Настолько маленького, что ни одна из вновь поступающих программ не может поместиться ни в одном из участков, хотя суммарный объем фрагментов может составить значительную величину, намного превышающую требуемый объем памяти.

**Методы распределения памяти с использованием дискового пространства**

1. **Понятие виртуальной памяти**

Уже достаточно давно пользователи столкнулись с проблемой размещения в памяти программ, размер которых превышал имеющуюся в наличии свободную память. Решением было разбиение программы на части, называемые *оверлеями*.

0-ой (нулевой) оверлей начинал выполняться первым.

Когда он заканчивал свое выполнение, он вызывал другой оверлей.

Все оверлеи хранились на диске и перемещались между памятью и диском средствами операционной системы. Однако разбиение программы на части и планирование их загрузки в оперативную память должен был осуществлять программист.

Развитие методов организации вычислительного процесса в этом направлении привело к появлению метода, известного под названием *виртуальная память*. Виртуальным называется ресурс, который пользователю или пользовательской программе представляется обладающим свойствами, которыми он в действительности не обладает.

**Виртуальная память** — это совокупность программно-аппаратных средств, позволяющих пользователям писать программы, размер которых превосходит имеющуюся оперативную память

Виртуальная память **решает следующие задачи**:

* размещает данные в запоминающих устройствах разного типа, например, часть программы в оперативной памяти, а часть на диске;
* перемещает по мере необходимости данные между запоминающими устройствами разного типа, например, подгружает нужную часть программы с диска в оперативную память;
* преобразует виртуальные адреса в физические.

Все эти действия выполняются *автоматически*, без участия программиста, то есть механизм виртуальной памяти является прозрачным по отношению к пользователю.

Наиболее распространенными реализациями виртуальной памяти является страничное, сегментное и странично-сегментное распределение памяти, а также свопинг.

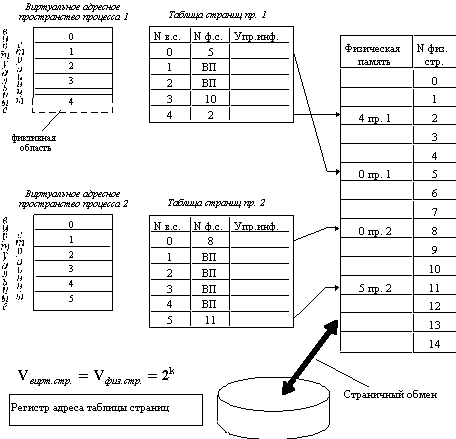
**Страничное распределение**

На рисунке 5 показана схема страничного распределения памяти. Виртуальное адресное пространство каждого процесса делится на части одинакового, фиксированного для данной системы размера, называемые *виртуальными страницами*. В общем случае размер виртуального адресного пространства не является кратным размеру страницы, поэтому последняя страница каждого процесса дополняется фиктивной областью.

Вся оперативная память машины также делится на части такого же размера, называемые *физическими страницами (или блоками).*

Размер страницы обычно выбирается равным степени двойки: 512, 1024 и т.д., это позволяет упростить механизм преобразования адресов.

При загрузке процесса часть его виртуальных страниц помещается в оперативную память, а остальные - на диск. Смежные виртуальные страницы не обязательно располагаются в смежных физических страницах. При загрузке операционная система создает для каждого процесса информационную структуру - таблицу страниц, в которой устанавливается соответствие между номерами виртуальных и физических страниц для страниц, загруженных в оперативную память, или делается отметка о том, что виртуальная страница выгружена на диск.



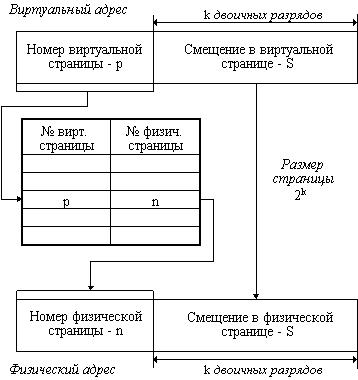
В данной ситуации может быть использовано много разных критериев выбора, наиболее популярные из них следующие:

* дольше всего не использовавшаяся страница,
* первая попавшаяся страница,
* страница, к которой в последнее время было меньше всего обращений.

После того, как выбрана страница, которая должна покинуть оперативную память, анализируется ее признак модификации (из таблицы страниц). Если выталкиваемая страница с момента загрузки была модифицирована, то ее новая версия должна быть переписана на диск. Если нет, то она может быть просто уничтожена, то есть соответствующая физическая страница объявляется свободной.

Рассмотрим механизм преобразования виртуального адреса в физический при страничной организации памяти (рисунок 8).

Виртуальный адрес при страничном распределении может быть представлен в виде пары (p, s), где p - номер виртуальной страницы процесса (нумерация страниц начинается с 0), а s - смещение в пределах виртуальной страницы. Учитывая, что размер страницы равен 2 в степени к, смещение s может быть получено простым отделением k младших разрядов в двоичной записи виртуального адреса. Оставшиеся старшие разряды представляют собой двоичную запись номера страницы p.



При каждом обращении к оперативной памяти аппаратными средствами выполняются следующие действия:

1. На основании начального адреса таблицы страниц (содержимое регистра адреса таблицы страниц), номера виртуальной страницы (старшие разряды виртуального адреса) и длины записи в таблице страниц (системная константа) определяется адрес нужной записи в таблице,
2. Из этой записи извлекается номер физической страницы,
3. К номеру физической страницы присоединяется смещение (младшие разряды виртуального адреса).

На производительность системы со страничной организацией памяти влияют временные затраты, связанные с обработкой страничных прерываний и преобразованием виртуального адреса в физический. При часто возникающих страничных прерываниях система может тратить большую часть времени впустую, на свопинг страниц. Чтобы уменьшить частоту страничных прерываний, следовало бы увеличивать размер страницы.

Страничное распределение памяти может быть реализовано в упрощенном варианте, без выгрузки страниц на диск. В этом случае все виртуальные страницы всех процессов постоянно находятся в оперативной памяти.

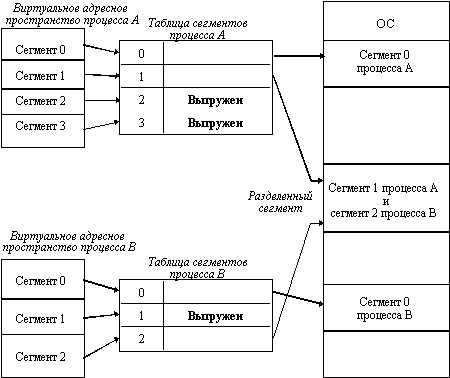
Такой вариант страничной организации хотя и не предоставляет пользователю виртуальной памяти, но почти исключает фрагментацию за счет того, что программа может загружаться в несмежные области, а также того, что при загрузке виртуальных страниц никогда не образуется остатков.

**Сегментное распределение**

При страничной организации виртуальное адресное пространство процесса делится механически на равные части. Это не позволяет дифференцировать способы доступа к разным частям программы (сегментам), а это свойство часто бывает очень полезным. Например, можно запретить обращаться с операциями записи и чтения в кодовый сегмент программы, а для сегмента данных разрешить только чтение. Кроме того, разбиение программы на "осмысленные" части делает принципиально возможным разделение одного сегмента несколькими процессами. Например, если два процесса используют одну и ту же математическую подпрограмму, то в оперативную память может быть загружена только одна копия этой подпрограммы.

Рассмотрим, каким образом сегментное распределение памяти реализует эти возможности (рисунок 7). Виртуальное адресное пространство процесса делится на сегменты, размер которых определяется программистом с учетом смыслового значения содержащейся в них информации. Отдельный сегмент может представлять собой подпрограмму, массив данных и т.п. Иногда сегментация программы выполняется по умолчанию компилятором.

При загрузке процесса часть сегментов помещается в оперативную память (при этом для каждого из этих сегментов операционная система подыскивает подходящий участок свободной памяти), а часть сегментов размещается в дисковой памяти. Сегменты одной программы могут занимать в оперативной памяти несмежные участки.



Система с сегментной организацией функционирует аналогично системе со страничной организацией: время от времени происходят прерывания, связанные с отсутствием нужных сегментов в памяти, при необходимости освобождения памяти некоторые сегменты выгружаются, при каждом обращении к оперативной памяти выполняется преобразование виртуального адреса в физический. Кроме того, при обращении к памяти проверяется, разрешен ли доступ требуемого типа к данному сегменту.

Виртуальный адрес при сегментной организации памяти может быть представлен парой (g, s), где g - номер сегмента, а s - смещение в сегменте. Физический адрес получается путем сложения начального физического адреса сегмента, найденного в таблице сегментов по номеру g, и смещения s.

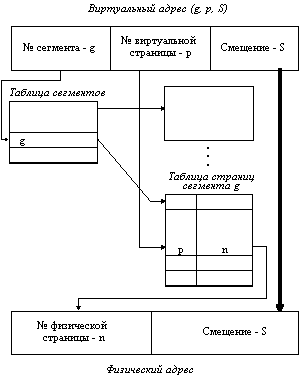
Недостатком данного метода распределения памяти является фрагментация на уровне сегментов и более медленное по сравнению со страничной организацией преобразование адреса.

При активизации очередного процесса в специальный регистр процессора загружается адрес таблицы страниц данного процесса.

**Странично-сегментное распределение**

Данный метод представляет собой комбинацию страничного и сегментного распределения памяти и, вследствие этого, сочетает в себе достоинства обоих подходов. Виртуальное пространство процесса делится на сегменты, а каждый сегмент в свою очередь делится на виртуальные страницы, которые нумеруются в пределах сегмента. Оперативная память делится на физические страницы. Загрузка процесса выполняется операционной системой постранично, при этом часть страниц размещается в оперативной памяти, а часть на диске. Для каждого сегмента создается своя таблица страниц, структура которой полностью совпадает со структурой таблицы страниц, используемой при страничном распределении. Для каждого процесса создается таблица сегментов, в которой указываются адреса таблиц страниц для всех сегментов данного процесса. Адрес таблицы сегментов загружается в специальный регистр процессора, когда активизируется соответствующий процесс.

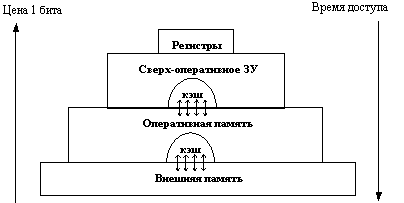
На рисунке 8 показана схема преобразования виртуального адреса в физический для данного метода.



**Иерархия запоминающий устройств, принцип кэширования данных**

Память вычислительной машины представляет собой иерархию запоминающих устройств (внутренние регистры процессора, различные типы сверхоперативной, оперативной памяти, диски, ленты). Отличающихся среднем временем доступа и стоимостью хранения данных в расчёте на один бит (рисунок 9)

Кэш память – это способ организации совместного функционирования двух типов запоминающих устройств, отличающихся временем доступа и стоимостью хранения данных, который позволяет уменьшить среднее время доступа к данным за счёт динамического копирования в “быстрое” запоминающее устройство (ЗУ) часто используемой информации из “медленного” ЗУ



В системах, оснащенных кэш-памятью, каждый запрос к оперативной памяти выполняется в соответствии со следующим алгоритмом:

Просматривается содержимое кэш-памяти с целью определения, не находятся ли нужные данные в кэш-памяти; кэш-память не является адресуемой, поэтому поиск нужных данных осуществляется по содержимому - значению поля "адрес в оперативной памяти", взятому из запроса.

На практике в кэш-память считывается не один элемент данных, к которому произошло обращение, а целый блок данных, это увеличивает вероятность так называемого "попадания в кэш", то есть нахождения нужных данных в кэш-памяти.

Пусть имеется основное запоминающие устройство со средним временем доступа к данным и кэш-память, имеющая время доступа , очевидно, что . Обозначим через t среднее время доступа к данным в системе с кэш-памятью, а через p -вероятность попадания в кэш. По формуле полной вероятности имеем:

Из нее видно, что среднее время доступа к данным в системе с кэш-памятью линейно зависит от вероятности попадания в кэш и изменяется от среднего времени доступа в основное ЗУ (при р=0) до среднего времени доступа непосредственно в кэш-память (при р=1).

## Лекция 8 - Основные понятия баз данных.

**Задачи, решаемые с помощью баз данных**

Информационная система представляет собой аппаратно-программный комплекс, обеспечивающий выполнение следующих функций:

* Ввод данных об объектах некоторой предметной области;
* Надёжное хранение и защита данных во внешней памяти вычислительной системы;
* Дополнение, удаление, изменение данных;
* Сортировка, выборка данных по запросам пользователей;
* Выполнение специфических для данной предметной области преобразований информации;
* Предоставление пользователям удобного интерфейса;
* Обобщение данных и составление отчётов.

Совокупность взаимосвязанных данных, называемых **структурой данных**.

Совокупность структурированных данных, всегда относящихся к какой-либо предметной области, называется **базой данных** (БД). То есть **БД** – организованная структура, предназначенная для хранения информации.

Совокупность программ, реализующих в БД функции информационные системы (ИС) в удобной для пользователя форме, называется **системой управления базой данных** (СУБД).

**СУБД** – комплекс программ, предназначенных для создания структуры новой базы, ведения данных и визуализации информации.

**Ведение** (***сопровождение, поддержка***) **данных** – термин, объединяющий действия по добавлению, удалению или изменению хранимых данных.

Под **визуализацией** **информации** **БД** понимается отбор отображаемых данных в соответствии с заданным критерием, упорядочение, оформление и последующая выдача на устройство вывода или передача по каналам связи.

**Классификация БД**

По технологии обработки данных БД подразделяются на *централизованные* и *распределённые*.

**Централизованная** **БД** хранится целиком в памяти одной вычислительной системы. Если система входит в состав сети, то возможен доступ к этой БД других систем.

**Распределённая** БД состоит из нескольких, возможно пересекающихся или дублирующих друг друга БД, хранимых в памяти разных вычислительных систем, объединённых в сеть.

По **способу доступа к данным БД** различают *локальный* (автономный) и *удалённый* (сетевой) доступ.

**Локальный** **доступ** предполагает, что СУБД обрабатывает БД, которая хранится на том же компьютере.

**Удалённый** **доступ** – это обращение к БД, которая хранится на одном из компьютеров, входящих в компьютерную сеть. Удалённый доступ может быть выполнен по принципу **файл-сервер** или **клиент-сервер**.

**Архитектура файл-сервер** (толстый клиент) предполагает выделение одного из компьютеров сети (сервер) для хранения централизованной БД. Все остальные компьютеры сети (клиенты) исполняют роль рабочих станций, которые копируют требуемую часть централизованной БД в свою память, где и происходит обработка. Однако при большой интенсивности запросов к централизованной БД увеличивается нагрузка на каналы сети, что приводит к снижению производительности ИС в целом.

**Транзакция** – элементарная операция по обработке данных, имеющая фиксированные начало, конец (успешное или неуспешное завершение) и ряд других характеристик.

**Архитектура** **клиент-сервер** (тонкий клиент) предполагает, что сервер, выделенный для хранения централизованной БД, дополнительно производит обработку клиентских запросов. Клиенты получают по сети уже обработанные данные. Учитывая широкое распространение БД в самых различных областях, в последнее время архитектура клиент-сервер применяется и на одиночных вычислительных системах.

**Многозвенная архитектура** предполагает наличие сервера приложений, на котором выполняется вся вычислительная работа, что позволяет уменьшить нагрузку на сервер БД.

**Распределённая архитектура** обладает ещё более высокой надёжностью, в которой вычислительная система состоит из нескольких компонентов, распределённых по разным серверам. Специальные программы-мониторы следят за корректностью работы каждого из компонентов и, при необходимости, запускают дублирующие компоненты на других компьютерах.

**Реляционная модель данных**

**Реляционная** (от англ. relation – отношение) **модель данных** является наиболее универсальной, к ней могут быть сведены другие модели (иерархическая и сетевая).

Важнейшим понятием реляционных моделей данных является сущность.

**Сущность** – это объект любой природы, данные о котором хранятся в БД. Для представления данных об объектах и их взаимосвязях используются *отношения*. Каждое **отношение –** этореляционная таблица. Каждый конкретный экземпляр сущности представляется совокупностью элементов строки, которая называется **кортежем** (или *записью*). Каждый столбец есть **домен** (альтернативные названия – *атрибут и поле*), по имени которого группируются данные различных экземпляров сущности. Строка заголовков называется **схемой отношения**. Количество доменов определяет **степень отношения**, количество кортежей – его **мощность**.

Каждая реляционная таблица должна обладать следующими свойствами:

* Один элемент таблицы – одни элемент данных;
* Все столбцы таблицы содержат однородные по типу данные (целочисленный, числовой, текстовый и т.д.);
* Каждый столбец имеет уникальное имя;
* Чисто столбцов задаётся при создании таблицы;
* Порядок записей в отношении может быть произвольным;
* Записи не должны повторяться;
* Количество записей в отношении не ограничено.

Структура простейшей базы, состоящей из одной таблицы, представлена *полями* (столбцами) и *записями* (строками). Даже если в базе нет ни одной записи (пустая база), это всё равно полноценная база – в ней содержится информация о методах хранения данных, хотя сами данные пока отсутствуют – её структура представлена набором полей.

*Первичным ключом* отношения называется поле или группа полей, однозначно определяющие запись.

Для связи реляционных таблиц необходимо ввести в обе таблицы одинаковые по типу поля, по которым определится связь между записями обеих таблиц.

Связи бывают несколько типов:

* *Один к одному* (1:1) – любая запись одной таблицы может быть связана только с одной записью другой и наоборот. По сути, каждая пара записей является одной записью, поля которой разделены на две таблицы. То есть часть полей находится в одной таблице, а оставшаяся часть – в другой, являющейся продолжением первой.
* *Один ко многим* (1:М или 1:∞ ) – любая запись одной таблицы может быть связана с несколькими записями другой, но любая запись второй таблицы связана только с одной записью первой таблицы.
* *Многие ко многим* (М:М или ∞:∞) – любая запись одной таблицы может связана с несколькими записями другой и наоборот. **В явном виде это связь может не поддерживаться***,* обычно она организуется путём создания дополнительных таблиц.

**Свойства полей базы данных (!!!!!)**

Рассмотрим основные свойства полей БД на примере СУБД Microsoft Access:

* Имя поля – идентификатор, по которому происходит обращение к данным этого поля при автоматических операциях с базой (используется в качестве заголовка по умолчанию);
* Подпись – определяет заголовок столбца, отличный от имени поля;
* Тип поля – определяет тип данных, содержащихся в данном поле;
* Обязательное поле – свойство, определяющее обязательность ввода данных;
* Пустые строки – в отличие от предыдущего свойства, разрешает ввод пустых строк для некоторых типов данных (например, текстовых);
* Размер поля – задаёт предельную длину (в символах) данных, которые могут размещаться в данном поле;
* Формат поля – определяет способ форматирования данных в ячейках, принадлежащих полю;
* Маска ввода – определяет форму, в которой вводятся данные в поле;
* Значение по умолчанию – автоматически вводимое в поле значение при создании новой записи;
* Условие на значение – ограничение, используемое для проверки правильности ввода данных;
* Сообщение об ошибке – текстовое сообщение, выдаваемое при попытке ввода ошибочных данных, если задано предыдущее свойство;
* Индексированное поле – свойство, ускоряющее операции поиска и сортировки записей по значениям данного поля. Возможна автоматическая проверка и исключение дублирования данных

**Типы данных**

* Текстовый – тип данных для хранения текста длинной до 255 символов;
* Поле МЕМО – тип данных для хранения больших объёмов текста (до 65535 символов). Физически текст хранится в другом месте базы данных, а в поле помещается только указатель на него;
* Числовой – тип данных для хранения чисел, формат представления которых (целые, действительные и др.) задаётся свойством Размер поля;
* Дата/время – тип данных для хранения календарных дат и времени;
* Денежный - тип данных для хранения денежных сумм. Денежный тип удобнее, чем специальная настройка формата числового типа и имеет некоторые особенности, например, округления;
* Счётчик – специальное поле для натуральных чисел с автоматическим наращиванием. Используется для естественной нумерации записей;
* Логический - тип данных для хранения логических величин (вкл/выкл);
* Поле объекта OLE – тип данных для хранения объектов OLE (например, мультимедийных). Как и поле МЕМО, содержимое хранится в специальном месте БД, иначе работа с базой была бы замедленной;
* Гиперссылка – тип данных для хранения URL адресов Web-объектов Интернета. При щелчке на ссылке запускается браузер, выполняющий загрузку и воспроизведение указанного объекта;
* Вложение – тип данных для хранения внешних файлов, появившийся в версии Access 2007;
* Вычисляемый – тип данных, появившийся в версии Access 2010.

**Безопасность и объекты баз данных**

Базы данных – это особые структуры. Информация, которая в них содержится, имеет большую, часто общественную ценность (учёт автомобилей в ГИБДД, обслуживание счетов в банках, обработка результатов ЕГЭ…) Нередко с одной и той же базой работают тысячи пользователей по всей стране. От информации, её актуальности и сохранности, может зависеть благополучие множества людей. Поэтому целостность содержимого базы не должна зависеть не от оператора, который может забыть сохранить ведённые данные, ни от перебоев в электропитании.

Проблема безопасности решается тем, что в СУБДЬ для сохранения данных используется двойной подход. С одной стороны, изменение структуры базы, добавление или удаление таблицы и другие, так сказать, глобальные операции происходят с сохранением файла базы данных. О таких операциях СУБД предупреждает пользователя (разработчика), их не проводят с базой, находящейся в эксплуатации.

С другой стороны, операции по наполнению и редактированию содержимого базы, не затрагивающие её структуру, максимально автоматизированы, выполняются без предупреждения, в обход операционной системы. Если мы добавим или удалим записи, отредактируем хотя бы одно поле, то все изменения тут же сохранятся на диске.

Каждая СУБД реализует свои типы объектов. Перечислим основные типы объектов, которые используются в БД на примере MS Access.

**Таблицы** – основные объекты любой реляционной БД. В таблицах хранится как структурная базы (поля, их типы и свойства), так и все данные (записи).

**Запросы** – объекты, предназначенные для доступа к нужным данным из одной или нескольких таблиц. Результат выполнения запроса также представляется в табличной форме. Особенность запросов на выборку состоит в том, что из нужных полей и записей базовых таблиц в оперативной памяти создаётся временная результирующая таблица, которую ещё называют моментальным снимком. Основное назначение запросов – обеспечение *удобства* и *безопасности.*

(!!!!) Удобство работы с запросом состоит в том, что, во-первых, пользователю предоставляется доступ не ко всем данным, а только к тем, которые удовлетворяют указанным в запросе критериям. Во-вторых, отобранные по запросу данные могут быть упорядочены по указанному в запросе принципу, а в таблицах они хранятся в *естественном* порядке по мере поступления. В-третьих, применяемые по запросу операции сортировки и фильтрации выполняются гораздо быстрее, чем с базовыми таблицами, так как результирующая таблица находится не во внешней, а в оперативной памяти.

**Формы** – это объекты, предназначенные для удобного и безопасного (аналогично запросам) *ввода* данных. В отличие от запросов, представляющих данные в виде таблиц, формы, представляют собой окна с размещёнными на них полями данных и элементами управления (кнопки, надписи, счётчики и прочие). К формам специальные средства оформления, наглядно отображающие существующие данные и облегчающие ввод новых. Преимущества форм можно представить, если ввод данных осуществляется с заполненных бумажных бланков. В этом случае графическое представление экранной формы повторяет оформление бланка – это заметно упрощает работу наборщика, снижает его утомление и уменьшает количество ошибок, пропущенных и неверно заполненных полей.

**Отчёты** – объекты, предназначенные только для *вывода* данных. Отчёты позволяют на основе таблиц и запросов сформировать наглядные документы, которые могут быть распечатаны или включены в документ другого приложения. В отчётах могут быть не только отфильтрованные и отсортированные, но и сгруппированные данные. Добавляются специальные элементы оформления, характерные для печатных документов: колонтитулы, номера страниц, время создания отчёта и т.п.

**Макросы** и **модули** – объекты, предназначенные для автоматизации работы с базой. Макросы состоят из последовательности внутренних команд СУБД и предназначены для автоматизации повторяющихся операций. *Модули* создаются путём программирования на языке Visual Basic for Applications(BBA) и предназначены для создания новых функций, реализации нестандартных возможностей, позволяющих удовлетворить специфические требования заказчика, повысить быстродействие системы управления и уровень её защищённости.

**Проектирование БД**

Проектирование БД представляет собой длительный, трудоёмкий, слабо формализованный процесс, от которого зависит жизнеспособность и эффективность проектируемой базы, её способность к развитию. Проектирование БД выполняется, как правило, коллективом разработчиков и включает следующие этапы:

* Анализ предметной области;
* Проектирование и кодирование;
* Тестирование и сопровождение.

*Анализ предметной области* необходим для составления технического задания на разработку базы данных.

*Проектирование* баз данных осуществляется на двух уровнях – *физическом* и *логическом*. На физическом уровне решаются вопросы размещения данных на внешних носителях. Во многом эта работа выполняется СУБД автоматически без участия разработчика. На логическом уровне создаётся структура базы, начиная с построения модели данных предметной области (инфологической, то есть информационно-логической модели) и заканчивания схемой данных (описанием таблиц и связей между ними).

## Лекция 9 - Основы информационной безопасности

**\_1\_ Угрозы информационной безопасности**

*Программные пакеты, дешифрование, шифрование, тестовые вирусные пакеты… на 2-ом курсе*

**Конфиденциальность** – свойство информации быть доступной только ограниченному кругу субъекту доступа.

**Целостность** – свойство информации сохранять свою структуру и содержание в процессе хранения, использования и передачи.

**Доступ** – возможность субъекта осуществлять определённые действия с информацией. При выполнении правил разграничения доступа – санкционированный (иначе – несанкционированный).

Под **угрозой информационной безопасности** понимается действие или событие, которое может привести к разрушению, искажению или несанкционированному использованию информационных ресурсов, включая хранимую, передаваемую и обрабатываемую информацию, а также программные и аппаратные средства. Делятся на **случайные/непреднамеренные** и **умышленные**.

**Пассивные** угрозы, как правило, направлены на несанкционированное использование информационных ресурсов, не оказывая при этом влияния на функционирование системы.

**Активные** угрозы имеют цель: нарушить нормальный процесс функционирования системы посредством целенаправленного воздействия на аппаратные, программные и информационные ресурсы.

К основным угрозам безопасности информации относят:

* Раскрытие конфиденциальной информации;
* Компрометация информации;
* Несанкционированное использование информационных ресурсов;
* Ошибочное использование информационных ресурсов;
* Несанкционированный обмен информацией;
* Отказ от информации;
* Отказ в обслуживании.

**Компрометация информации**, как правило, реализуется посредством внесения несанкционированных изменений в базы данных, в результате чего её потребитель вынужден либо отказаться от неё, либо предпринимать дополнительные усилия для выявления изменений и восстановления истинных сведений.

**Несанкционированное использование** информационных ресурсов, с одной стороны, является средством раскрытия или компрометации информации, а с другой – имеет самостоятельное значение, поскольку, даже не касаясь пользовательской или системной информации, может нанести определённый ущерб абонентам и администрации.

**Ошибочное использование** информационных ресурсов, будучи санкционированным, тем не менее, может привести к разрушению, раскрытию или компрометации указанных ресурсов.

**Несанкционированный обмен** информацией между абонентами может привести к получению одним из них сведений, доступ к которым ему запрещён, что по своим последствиям равносильно раскрытию содержания банковской информации.

**Отказ от информации** состоит в непризнании получателем или отправителем этой информации фактов её получения или отправки.

**Отказ в обслуживании** представляет собой весьма существенную и распространённую угрозу, источником которой является сама система и применяемые в ней технологии.

**\_2\_ Методы и средства защиты информации**

Создание базовой системы защиты информации основывается на следующих принципах:

* **Комплексный подход** к построению системы защиты при ведущей роли организационных мероприятий, означающий оптимальное сочетание программных аппаратных средств и организационных мер защиты и подтверждённый практикой создания отечественных и зарубежных систем защиты;
* **Разделение** и **минимизация** полномочий по доступу к обрабатываемой информации и процедурам обработки т.е. предоставление пользователям минимума строго определённых полномочий, достаточных для успешного выполнения ими своих служебных обязанностей, с точки зрения автоматизированной обработки доступной им конфиденциальной информации;
* **Полнота контроля** и регистрации попыток несанкционированного доступа, т.е. необходимость точного установления идентичности каждого пользователя и протоколирования его действий для проведения возможного расследования, а также невозможность совершения любой операции обработки информации в системе без её предварительной регистрации;
* **Обеспечения надёжности** системы защиты, т.е. невозможность снижения уровня надёжности при возникновении в системе сбоев, отказов, преднамеренных действий нарушителя или непреднамеренных ошибок пользователей и обслуживающего персонала;
* **Обеспечение контроля** за функционированием системы защиты, т.е. создание средств и методов контроля работоспособности механизмов защиты;
* **«Прозрачность»** системы защиты информации для общего, прикладного программного обеспечения и пользователей;
* **Экономическая целесообразность** использования системы защиты, выражающаяся в том, что стоимость разработки и эксплуатации систем защиты информации должна быть меньше стоимости возможного ущерба, наносимого объекту в случае его разработки и эксплуатации без системы защиты информации;
* Следует отметить, что какими бы совершенными не были программно-технические средства защиты информации от несанкционированного доступа, без надлежащей организационной поддержки и точного выполнения предусмотренных проектов документацией процедур проблему обеспечения безопасности информации в должной мере не мере

Рассмотрим методы и средства обеспечения безопасности информации, составляющие основу механизмов защиты.



* **Препятствие** – метод физического преграждения пути злоумышленнику к защищаемой информации (к аппаратуре, носителям информации и т.д.)
* **Управление доступом** – метод защиты информации регулированием использования всех ресурсов компьютерной информационной системы (элементов баз данных, программных и технических средств)

Управление доступом включает следующие функции:

* Идентификация пользователей, персонала и ресурсов системы (присвоение каждому объекту персонального идентификатора);
* Аутентификация (опознание, установление подлинности) объекта или субъекта по предъявленному им идентификатору;
* Проверка полномочий (проверка соответствия дня недели, времени суток, запрашиваемых ресурсов и процедур установленному регламенту);
* Разрешение и создание условий работы в пределах установленного регламента;
* Регистрация (протоколирование) обращений к защищаемым ресурсам;
* Реагирование (сигнализация, отключение, задержка работ, отказ в запросе) при попытках несанкционированных действий.
* **Маскировка** – метод защиты информации путём её криптографического закрытия. Этот метод защиты широко применяется за рубежом как при обработке, так и при хранении информации, в том числе на дискетках. При передаче информации по каналам связи большой протяженности этот метод является единственно надёжным;
* **Регламентация –** метод защиты информации, создающий такие условия автоматизированной обработки, хранения и передачи защищаемой информации, при которых возможности несанкционированного доступа к ней сводились бы к минимуму.
* **Принуждение –** такой метод защиты, при котором пользователи и персонал системы вынуждены соблюдать правила обработки, передачи и использования защищаемой информации под угрозой материальной, административной или уголовной ответственности.
* **Побуждение –** такой метод защиты, который побуждает пользователя и персонал системы не разрушать установленные порядки за счет соблюдения сложившихся моральных и этических норм (как регламентированных, так и неписаных).
* Рассмотренные методы обеспечения безопасности реализуются на практике за счет применения различных средств защиты, таких как технические, программные, организационные, законодательные и морально-этические.

**Основные средства защиты**

Основные средства защиты делятся на **формальные** (выполняющие защитные функции строго по заранее предусмотренной процедуре без непосредственного участия человека) и **неформальные** (определяются целенаправленной деятельностью человека либо регламентируют эту деятельность).

**К формальным средствам относятся следующие:**

* **Технические** средства реализуются в виде электрических, электромеханических и электронных устройств. Вся совокупность технических средств делится на физические и аппаратные.
* **Физические** средства реализуются в виде автономных устройств и систем. Например, замки на дверях, где размещена аппаратура, решетки на окнах, электронно-механическое оборудование охранной сигнализации.
* Под **аппаратными техническими** средствами принято понимать устройства, встраиваемые непосредственно в вычислительную технику, или устройства, которые сопрягаются с подобной аппаратурой по стандартному интерфейсу.
* **Программные** средства представляют из себя программное обеспечение, специально предназначенное для выполнения функций защиты информации.

**Неформальные средства:**

* **Организационные** средства защиты представляют собой организационно-технические и организационно-правовые мероприятия, осуществляемые в процессе создания и эксплуатации вычислительной техники, аппаратуры телекоммуникаций для обеспечения защиты информации. Организационные мероприятия охватывают все структурные элементы аппаратуры на всех этапах их жизненного цикла (строительство помещений, проектирование компьютерной информационной системы банковской деятельности, монтаж и наладка оборудования, испытания, эксплуатация).
* **Законодательные** средства защиты определяются законодательными актами, которыми регламентируются правила пользования, обработки и передачи информации ограниченного доступа и устанавливаются меры ответственности за нарушение этих правил.
* **Морально-этические** средства защиты реализуются в виде всевозможных норм, которые сложились традиционно или складываются по мере распространения вычислительной техники и средств связи в обществе. Эти нормы большей частью не являются обязательными как законодательные меры, однако несоблюдение их ведет обычно к потере авторитета и престижа человека.

## Лекция 10 - Основы программирования в Python

Python интерпретируемый язык с динамической типизацией (приём, используемый в языках программирования и языках спецификации, при котором переменная связывается с типом в момент присваивания значения, а не момент объявления переменной)

Плюсы Python:

* Полностью open-source (открытый исходный код)
* Расширяемость за счёт множества библиотек
* Очень широкая сфера применения
* Приятный синтаксис

Сфера применения:

* Анализ данных и машинное обучение
* Работа с xml/html файлами, http-запросами, протоколом ftp
* Разработка веб-приложений, веб сценариев
* Системное администрирование
* Разработка десктопных приложений
* Gamedev
* Программирование математических и научных вычислений
* Роботехника и пр.

Специальная программа-интерпретатор позволяет выполнять команды языка средствами интерактивной оболочки. Разработаны и применяются также инструменты компиляции Python-программ.

Многие средства для работы с Python являются кросс-платформенными, что позволяет легко переносить программы из одной среды функционирования в другую.

Осуществлена поддержка многобайтной кодировки Unicode.

На Python написано большое количество популярных программ

* BitTorrent
* Ubuntu Software Center – свободное ПО для поиска, установки и удаления пакетов в системе Ubuntu Linux
* Blender – свободный, профессиональный пакет для создания трёхмерной компьютерной графики
* GIMP

## Средства разработки на Python

Существуют средства, облегчающие процесс создания программ на Python:

* Специализированные лексические анализаторы
* Редакторы для программистов
* Интегрированные среды разработки: PyCharm, Eric, Geany и др.
* Платформы Anaconda, ActivePython, Python(x,y)

### Лидирующей платформой разработки является Anaconda

Это высокопроизводительное расширение Python, которое включает более 100 популярных языков программирования, менеджер пакетов conda, интерактивную оболочку IPython и графический редактор Spyder

Anaconda позволит работать без риска случайного повреждения ПО компьютера, от которого зависит его функциональность, выбрать языковую версию и устанавливать пакеты расширения без побочных эффектов

### Интегрированная среда разработки EDLE Python

IDLE (Integrated DeveLopment Environment) – это интегрированная среда разработки на языке Python, созданная с помощью кросс-платформенной графической библиотеки на основе средств Tkinter (TL – toolkit)

Среда разработки EDLE Python включает интерактивную оболочку (Python Shell) и редактор кода

Основное назначение оболочки – отладка и выполнение (прогон) программы. В строке, начинающейся символами приглашения >>>, можно записать команду языка, которая будет исполнена гепосредственно после нажатия Enter, иначе говоря, в интерактивном режиме.

Однако реализовать полноценный алгоритм и написать программу в оболочке EDLE Python невозможно. Для этого используется редактор кода, вызываемый из меню File (файл) интерактивной оболочки

Программа на языке Python – это последовательность команд, которые чаще всего обрабатываются интерпретатором сверху вниз. Считывая очередную команду, интерпретатор сразу её выполняет, не переводя программу на машинный код. Исключения составляют: выполнение альтернативных блоков.

Функция прерывает линейную последовательность инструкций, приостанавливая выполнение основной программы, и контроль передается на выполнение функции. По окончании работы функции контроль снова возвращается в основную программу.

**Линейная программа** – это реализация на языке программирования алгоритмической конструкции, называемой *следование, линейный алгоритм и т.п.* – набор команд, выполняемых во времени строго последовательно друг за другом.

Как правило, линейная программа включает:

* Операторы присваивания – непосредственное указание значений переменных в программном коде
* Операторы ввода – предоставление пользователю возможности ввести любое значение во время выполнения программы
* Операторы вывода – отображение на экране или запись в файл значений переменных и выражений

Язык программирования является объектно-ориентированным. В нем все данные – числа, логические значения, строки, структуры данных, переменные, функции – реализованы как **объекты.** Таким единым представлением обеспечивается *стабильность языка.*

**Объект** – абстракция для данных, обладающая:

* *Идентичностью* (специфическая опознавательными признаками)
* *Значением* – содержит фрагмент данных
* *Типом*, определяющим допустимые операции с этими данными

Python является строго типизированным языком программирования. Это означает, что идентичность и тип объекта изменить нельзя.

Как упоминалось ранее, переменные относятся к *объектам* Python, которые создаются при использовании оператора присваивания и не требуют предварительного описания. По сути *переменная* – идентификатор, который ссылается на значение в памяти компьютера, а *имя объекта* – это ссылка на объект, а не сам объект. Таким образом, присваивание не копирует значение в ячейку с определенным именем, а добавляет имя к объекту, который содержит данные.

При использовании переменных в программе следует учитывать **требования к именам переменных:**

* Имена составляются их строчных и прописных букв, включая символ подчеркивания «\_» и цифры
* Не должны начинаться с цифры
* Являются регистро-чувствительными. Это означает, что переменные с именами «x» и «X» - два разных объекта
* Зарезервированные слова Python не могут использоваться в качестве имен переменных

Программы на языках высокого уровня, в том числе на Python, в основном оперируют данными, которые хранятся как переменные.

**Переменная –** это объект, характеризуемый определенным типом, имеет имя и значение, которое может изменяться во время выполнение программы. Другими словами, переменная – это имя для зарезервированного места в памяти компьютера, предназначенного для хранения значений некоторого типа.

**Тип данных –** характеристика набора данных, определяющая:

* Диапазон возможных значений и их представление
* Допустимые над данными значениями операции
* Способ хранения этих значений в памяти

…..

К *стандартным* (базовым) типам данных Python относятся:

* Числа (numbers)
* Строка (string)
* Список (list)
* Кортеж (Tuple)
* Множество (set)
* Словарь (dictionary)

*Числовой тип данных* предназначен

…

**Комплексные числа** (complex) создаются путем добавления символа “j” в конце

Преобразование типов числовых данных производится автоматически, при этом область значений результирующего типа данных всегда равна или шире, чем область значений операндов. Например, при сложении двух целых чисел результат – целое число, а при сложении целого и вещественного числа получается вещественный результат.

Встроенные функции

|  |  |
| --- | --- |
| Функции | Описание |
| abs(x) | Возвращает абсолютное значение (модуль) целого, вещественного или действительную часть комплексного |
| pow(x, y) |  |
| divmod(x,y) |  |
| round(x[,n])  type(x) |  |
| int(x) |  |
| float(x) |  |
| complex(x) |  |
| bin(x) |  |
| oct(x) |  |
| hex(x) |  |
| round(x[,n]) |  |

Операторы присваивания

|  |  |  |  |
| --- | --- | --- | --- |
| Оператор | Пояснение | Пример | Аналог |
| = | Присваивает левому операнду значение правого |  |  |
| += |  |  |  |
| -= |  |  |  |
| \*= |  |  |  |
| /= |  |  |  |

## Вывод информации на экран

Для вывода информации на экран в программах на языке Python используется инструкция print(), в скобках указываются: текст, число, имя переменной величины, выражение, метод (функция)

## **Лекция 11 - Вывод и ввод информации в среде программирования Python**.

# Программирование разветвляющихся алгоритмов.

## **Организация ввода данных**

Ввод данных непосредственно в ходе выполнения программы осуществляется с помощью функции input([prompt]), которая считывает вводимые символы с клавиатуры или файла и преобразует их в строку. Концом ввода данных является символ новой строки.

Функция input() имеет единственный необязательный аргумент **prompt**, при этом:

* Если аргумент prompt присутствует, то он выводится без перехода на новую строку
* Как правило, аргумент prompt – строка-приглашение ввода. В этом случае аргумент берется в кавычки
* Аргумент prompt изредка может быть переменной (используется без кавычек). Тогда выводится значение этой переменной

**Ввод чисел с помощью функции input()** реализуется двумя основными способами:

1. Преобразование введённой строки с помощью функций-конструкторов int(), float(), complex() и др
2. Применение спецфункции eval(), используемой для извлечения значений

Введенная строка (class ‘str’) преобразуется в целочисленное значение посредством функции-конструктора int(), после чего с ней можно выполнять арифм. Операции

Возможно преобразование введенной строки в числовое значение непосредственно в операторе присваивания

Для вывода информации во время выполнения программы используют функцию:

Print(objects, sep=’’, end=’\/, file=sys.stdout, flush=False),

Где objects – не ключевые аргументы, перечисляются через запятую; sep, end, file, flush - ключевые аргументы, при отсутствии которых или значении None используются значения по умолчанию.

Во многих других функциях Python присутствуют аргументы двух типов:

* Ключевые – имеющие имя и строго определенное назначение, изначально заданные в системе разработчиками
* Не ключевые – полностью зависят от пользователя-программиста

В функции print() перечисляются через запятую любые строки, константы других типов данных и/или имена переменных, значения которых необходимо вывести на экран или во внещний файл, - это не ключевые аругменты

Ключевые аргументы:

* Sep – строка, определяющая разделитель меду выводимыми объектами. По умолчанию данные выводятся, разделенные пробелом.
* End – строка, завершающая вывод данных. По умолчанию это перевод на новую строку («\n»)
* File – это текстовой поток, в который помещаются выводимые объекты
* Flush – аргумент, означающий, должен ли поток быть очищен после выполнения операции

Поток в программировании – абстрактное понятие, условно означающее некоторое множество перемещаемых данных. По умолчанию данные выводятся на экран и отображаются в окне оболочки Python. В качестве значения аргумента file можно указать имя внешнего файла, в который будет записана выводимая информация

## Форматированный вывод с помощью инструкции print()

Операция деления дает вещественный результат даже в случаях, когда она проводится над целыми числами

Количество цифр в дробной части можно ограничить. Для этого в инструкции print() перед выводимым значением следует записать точку, количество цифр в дробной части и букву f в виде:

Print(‘%.3f’% <значение>)

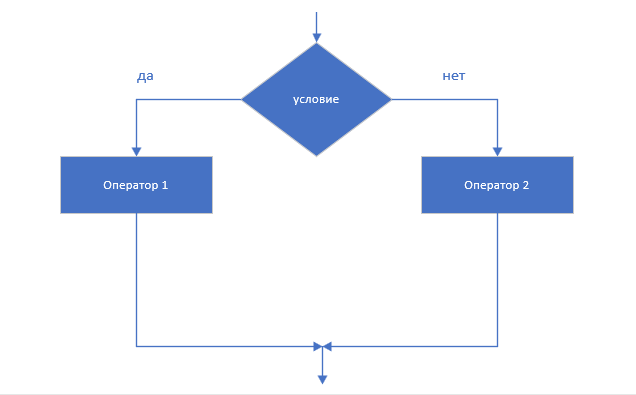
Также можно установить общее количество позиций экрана для вывода числа. Это количество ОбщКол указывается в формате вывода

Print(‘%7.3f’% <значение>)

Значение ОбщКол включает в себя цифры дробной части (КолДробн), разделитель-точку и знак числа

Ветвление – конструкция, которая дает возможность выбора только одного варианта из двух или более предложенных альтернатив.

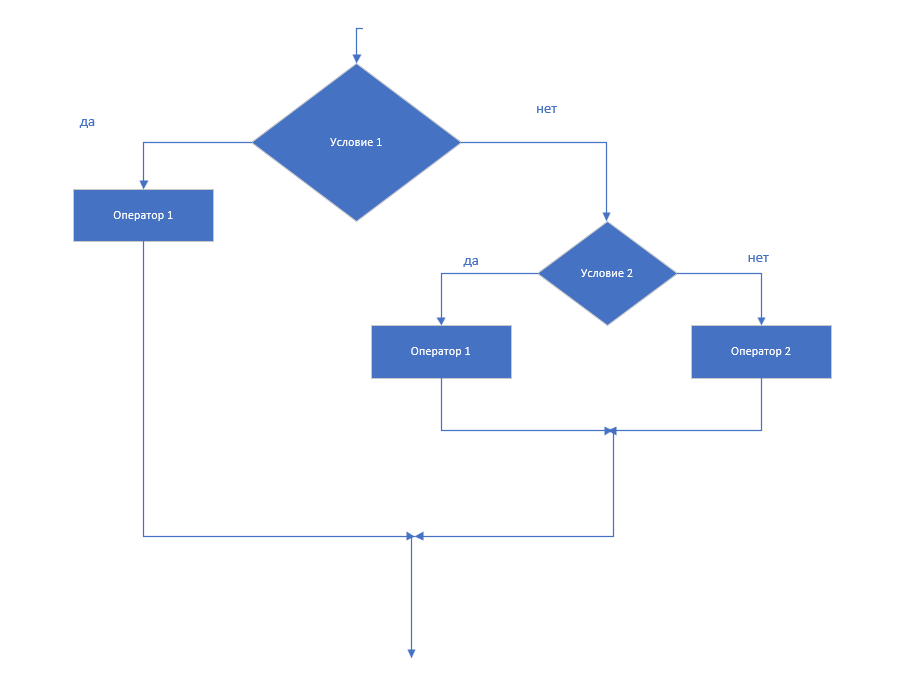
Принцип действия: проверяется условие. Если условие верно, то выполняется оператор 1, иначе – оператор 2. Далее осуществляется переход к следующему шагу алгоритма.



Неполное ветвление – алгоритмическая конструкция, в которой действие выполняется только в при истинности предложенного условия.

Принцип действия: проверяется условие

Вложенное ветвление – алгоритмическая констукция, в которой действие выполняется только при истинности (или ложности) некоторой последовательности предложенных условия



Множественное ветвление – алгоритмическая конструкция, позволяющая сделать выбор из произвольного числа предложенных, равнозначных по сути альтернатив.

## Лекция 12 - Разработка графического интерфейса в среде программирования Python

Пользователь может вводить информацию в спец. Поля, щелкать на тех или иных кнопках, выбирать опред. Значения в списках и т.п. Такую удобную пользователю среду называют графическим пользовательским интерфейсом (GUI)

Стандартной библиотекой в Python является tkinter. С помощью ее создаваемые элементы взаимодействия называются виджетами.

Основные виджеты: окно программы, кнопка, надпись, поле для ввода, флажки, переключатели, список, рамки.

Создание виджетов:

1. Импорт библиотеки tkinter в программу.

Import thinter from thinter \*

Или

From tkinter import \*

1. Указание инструкции, создающие окно программы
   1. Инструкция, объединяющая создание окна (чаще всего имя окна – root):

<имя окна> = Tk()

Инструкция, задающая свойства окна в виде:

<имя окна>.<имя свойства> = <значение свойства>

# root.title(‘Пример приложения’)

* 1. Размеры и положение на экране

Root.geometry(‘500 x 400 + 300 + 200’)

Здесь указаны размеры окна (ширина и высота) в пикселях и положение на экране его левого верхнего угла (задается от левого верхнего угла экрана по горизонтали и по вертикали в пикселях). Если размеры окна не указать, то они будут определяться размерами всех его виджетов

* 1. Root.resizable(False, False) #размер окна не может быть изменен

1. Общий вид инструкции для создания виджета:

<имя виджета> = <класс виджета>

\*\*\*

Классы для основных виджетов:

|  |  |
| --- | --- |
| **Виджет** | **Название** |
| Надпись | Label |
| Кнопка | Button |
| Поле для ввода | Entry |
| Флажок | Checkbutton |
| Переключатель | Radiobutton |
| Рамка | Frame |

Пример создания надписи, размещаемой в главном окне программы, с указанием двух свойств (выводимого в окне текста и используемого при этом шрифта):

Nadp = Label(root, text = ‘Характеристики шрифта’, font = ‘Arial 18’)

Инструкция по созданию кнопки с надписью “расчет” может выглядеть так:

Knopka\_raschet = Button(root, text = ‘Расчёт’)

Пример создания поля ввода:

pole = Entry(root, width = 20)

При описании указано свойство width – ширина поля (количество символов, которые могут быть введены на поле)

Как правило, рядом с полем ввода располагают надпись, информирующую пользователя о том, какое значение должно быть введено поле или какое значение выводится. Одним из свойств поля ввода является также свойство textvariable

Флажки создаются так:

Fl\_girn = Checkbutton(root, text = ‘Полужирный шрифт’, onvalue = 1, offvalue = 0)

Свойство text определяет текст, выводимый справа от флажка. Свойства onvalue и offvalue задают значения, которые будут возвращаться виджетом соответственно при отмеченном и неотмеченном флажке. По умолчанию флажки при запуске программы не отмечены.

Переключатели описываются следующим образом:

Per1 = Radiobutton(root, text = ‘Мужчина’, value = 1)

Per2 = Radiobutton(root, text = ‘Женщина’, value = 2)

Здесь значение свойства value для переключателя, включенного пользователем, также будет использоваться в программе…

Виджет список – это перечень некоторых элементов, из которого пользователь может выбирать один элемент или несколько. Инструкция для создания списка с одним выбором:

Spisok = Listbox(root, height = 4, width = 20, selectmode = SINGLE)

Заполнить этот виджет-список можно данными из “обычного” списка python:

Goroda = [‘Москва’, ‘Санкт-Петербург’, ‘Саратов’, ‘Омск’, ‘Тула’]

For el in goroda:

Spisok.insert(END,el)

Виджет-рамка создается так:

Ramka1 – Frame(root, width = 500, height = 100)

При выполнении программы рамка изображена не будет, так как по умолчанию её цвет совпадает с цветом окна.

Ramka1 = Frame(root, width = 500, height = 100, bg = ‘darkred’)

Для размещения виджетов в окне программы используются спецметоды. Их называют упаковщиками, или менеджерами расположения.

Самый просто способ – использование метода pack()

Как правило, этот упаковщик используют для размещения небольшого количества виджетов друг за другом. Например, если после описания кнопки с именем knopka1 в программе указать инструкцию для ее размещения кнопки в окне программы:

Knopka1.pack()

Если же нужно, чтобы виджеты располагались рядом по горизонтали, то при размещении кнопок следует указать аргумент side метода pack(), определяющего, к какой стороне оставшейся части окна должна примыкать каждая кнопка:

Knopka1.pack(side = ‘left’)

Knopka2.pack(side = ‘left’)

Упаковщик grid() позволяет разместить виджеты в виде таблицы. Для этого при размещении каждого виджета следует указать его “координаты” – номер строки таблицы row и номер столбца column:

Knopka1.grid(row = 0, column = 0)

Knopka2.grid(row = 0, column = 1)

Knopka3.grid(row = 1, column = 0)

Knopka4.grid(row = 1, column = 1)

Place() – позволяет размещать виджет в любом месте с любыми размерами. При использовании этого упаковщика необходимо указывать координаты левого верхнего угла каждого виджета от левого верхнего угла окна программы или другого родительского виджета.

Knopka.place(x = 0, y = 0)

===

Доступ к значениям в виджетах:

1. Get(). Если имя пола ввода vvod\_radiusa, то для того, чтобы в программе присвоить введенно значение пременной Radius, следует написать:

Radius = float(vvod\_radiusa.get())

1. Второй способ заключается в использовании связанной переменной, которая хранит нужное значение. При создании поля ввода следует описать такую переменную:

<Имя переменной> = StringVar()

<Имя переменной> = IntVar()

<Имя переменной> = DoubleVar() (вещественные значения)

<Имя переменной> = BooleanVar()

R = DoubleVar()

Vvod\_radiusa = Entry(root, width = 10, textvariable = r)

## Лекция 13 - Циклические вычислительные конструкцию. Обработка исключений. Текстовые строки в Python

Особенностью большинства вычислительных процессов является тот факт, что отдельные этапы вычислений повторяются многократно, при этом используются новые значения при каждом последующем вычислении.

Повторяющийся этап вычислений называют телом цикла, а вычислительный процесс – **циклическим**. Обычно под словом **цикл** понимают повторяющиеся действия или многократно выполняемую последовательность инструкций, организованную любым способом.

**Цикл в языках программирования высокого уровня** – это разновидность управляющей конструкции, предназначенная для организации многократного выполнения набора инструкций.

Различают **два основных вида** циклов:

• с известным числом повторений;

• с неизвестным числом повторений.

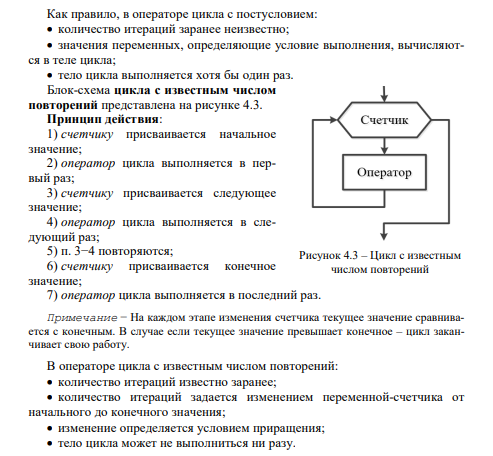
Циклы с известным числом повторений. Иначе их называют **арифметические** циклы. Как правило, имеют параметр цикла, характеризующийся начальным и конечным значением. Условием окончания арифметического цикла является достижение параметром цикла значения, большего конечного.

Циклы с неизвестным числом повторений (**итерационные** циклы). **Итерационным** называется вычислительный процесс, в котором для определения последующего значения переменной используется ее предыдущее значение. В частности, на итерационных циклах основан метод последовательных приближений.

Циклические алгоритмические конструкции можно подразделить на три основных вида:

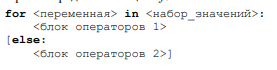
* с предусловием (итерационный);
* постусловием (итерационный);
* известным числом повторений (арифметический)





Параметрический цикл!!!

Цикл с известным числом повторений на языке Python На языке Python для реализации алгоритмической конструкции с известным числом повторений предназначен цикл for:



В отличие от других языков программирования, цикл for обладает более широкими возможностями на Python. В качестве параметра может использоваться:

* диапазонный объект
* последовательность (список, строка и др.)
* массив объектов;
* пользовательская структура данных.

Цикл for проходит по любому итерируемому объекту и на каждом шаге выполняет тело цикла. Использование в конструкции блока, предваряемого ключевым словом else, необязательно, но применяется для инициализации окончания работы цикла. В частности, удостоверяет, что все итерации цикла были выполнены полностью, т. е. цикл не был прерван принудительно

## Диапазонные объекты

На языке Python имеется специальная встроенная функция range(), которая генерирует диапазонные объекты – наборы значений, изменяющихся с некоторым постоянным шагом:

* range(N) – последовательность чисел от 0 до (N – 1) включительно. Наиболее часто находит применение для обхода индексов массива. Если N <= 0, то последовательность будет пустой;
* range(A, B) – последовательность чисел от A до (B – 1) включительно. Если B <= A, то последовательность будет пустой;
* range(A, B, Step), где A, B, Step целые числа – последовательность чисел от A до B (B не включается) с шагом Step. Если A == B или знаки шага Step и выражения (B − A) не совпадают, то последовательность будет пустой

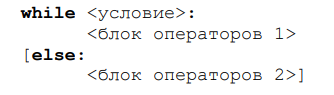
Встроенная функция range() может быть применена только если границы диапазона изменения переменной и шаг изменения – целые числа. Если требуется сгенерировать набор числовых значений с дробной частью, то необходимо использовать функцию arange(A, B, Step), входящую в состав модуля numpy.

## Цикл с предусловием

Цикл с предусловием относится к итерационным циклам. Итерационным называется вычислительный процесс, в котором для определения последующего значения переменной используется ее предыдущее значение. Итерационные циклы реализуют метод последовательных приближений, применяемый

* для нахождения предела последовательности;
* отделения и уточнения корней уравнения;
* вычисления значения интеграла и в некоторых других задачах.

На языке Python итерационный цикл с предусловием реализуется конструкцией следующей общей формы:



## Пропуск итераций и прерывание цикла

Довольно часто встречаются ситуации, когда в зависимости от текущих значений переменных, задействованных в теле цикла или являющихся его параметрами, требуется игнорировать следующий шаг цикла или досрочно закончить его выполнение. Оператор continue в конструкции

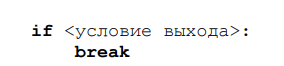


размещаемой внутри цикла, прерывает его с текущей позиции и передает управление:

* на следующую проверку условия оператора while;
* на следующую итерацию оператора for.

Иначе говоря, инструкция continue начинает следующий проход цикла, минуя оставшееся тело цикла. Пропуск итерации в операторе с предусловием while реализуется аналогично

Еще один оператор, позволяющий повлиять на поведение цикла – break в конструкции



размещаемой внутри цикла, прерывает его с текущей позиции и передает управление на оператор, следующий непосредственно после цикла. Прерывание break чаще всего используется для организации итерационного цикла с постусловием

## Текстовые строки в Python

Текстовые данные в Python представимы строковыми объектами (str) или просто строками (strings). Текстовая строка − неизменяемая последовательность символов в кодировке Unicode

Способы задания значений величинам строкового типа:

* С помощью инструкции присваивания

Im = ‘Дмитрий’

* С помощью инструкции input()

St = input(‘Введите строку символов’)

При хранении значения в памяти компьютера символы строки расположены подряд (в соседних ячейках).

