ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ:

ОСНОВНАЯ **ЦЕЛЬ** – ФОРМИРОВАНИЕ ПРОФЕССИОНАЛЬНОЙ КОМПЕТЕНТНОСТИ БУДУЩИХ СПЕЦИАЛИСТОВ В ОБЛАСТИ ПРИНЦИПОВ ПОСТРОЕНИЯ АЛГОРИТМОВ, СТРУКТУРНОГО ПРОЕКТИРОВАНИЯ И МЕТОДОВ РАЗРАБОТКИ ПРОГРАММ.

ЗАДАЧА ДИСЦИПЛИНЫ – РАЗВИТИЕ У УЧАЩИХСЯ АЛГОРИТМИЧЕСКОГО МЫШЛЕНИЯ, ФОРМИРОВАНИЕ ЗНАНИЙ О СВОЙСТВАХ АЛГОРИТМОВ И ПРИОБРЕТЕНИЕ ПРАКТИЧЕСКИХ НАВЫКОВ РАЗРАБОТКИ ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА ПРОГРАММИРОВАНИЯ ***PASCAL.***

КЛАССИФИКАЦИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ:

- ***Машинные языки***
- ***Языки ассемблера***
- ***Языки высокого уровня***

МАШИННЫЙ ЯЗЫК -

- это «природный язык» определенного компьютера.
- Он определяется при проектировании аппаратных средств этого компьютера.
- Машинные языки содержат строки чисел (в конечном счете сокращенные до единиц и нулей), которые являются командами компьютеру на выполнении большинства элементарных операций.
- Машинные языки ***машинно-зависимы*** (каждый машинный язык может быть использован только на компьютере одного определенного типа).
- Машинные языки тяжелы для человеческого восприятия:
- **+1300042774 +1400593419 +1200274027**

ЯЗЫКИ АССЕМБЛЕРА -

- **Очевидно** - программирование на машинных языках слишком медленно и утомительно для программистов.
- ***Языки ассемблера -*** похожие на английский язык аббревиатуры для представления элементарных компьютерных операций.
- Для преобразования программ на языке ассемблера в машинный язык разработаны ***программы трансляции -ассемблерами.***
- Фрагмент программы на языке ассемблера :
- **LOAD BASEPAY**
- **ADD OVERPAY**
- **STORE GROSSPAY**
- Такой код непонятен компьютеру до тех пор, пока не будет преобразован в компьютерный код.

ЯЗЫКИ ВЫСОКОГО УРОВНЯ -

- Для ускорения процесса программирования разработаны ***языки высокого уровня*** , в которых иногда достаточно написать один оператор для решения реальной задачи.
- Программы трансляции, которые преобразуют программы на языках высокого уровня в машинные коды, называются ***компиляторами***.
- Пример:
- **grossPay = basePay + overTimePay**
- Языки высокого уровня гораздо удобнее с точки зрения программистов по сравнению с языками ассемблера и с машинными кодами.
- Наиболее мощные и распространенные языков высокого уровня - ***Pascal, C (C++)*** и ***Basic***.

- Для непосредственного выполнения программ на языке высокого уровня без необходимости их компиляции в машинный язык были разработаны программы ***интерпретаторы***.
- Хотя скомпилированные программы выполняются быстрее чем интерпретируемые, интерпретаторы популярны, когда программы часто перекомпилируются для добавления в них новых возможностей и исправления ошибок.
- Но когда программа разработана, ее скомпилированная версия будет выполняться более эффективно.

- Появление языков высокого уровня не означает отказа от ***ассемблера*** и ***машинных кодов***.

- ***Ассемблер*** используется в качестве вставок для языков высокого уровня, а ***машинные коды*** используются для написания того, что не может сделать компилятор (и для написания самого компилятора).

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ (ООП) -

- Следующей ступенью стало ***объектно-ориентированное программирование (ООП)***. Язык ***Pascal*** в ***Object Pascal.***
- ***C*** превратился в ***C++.***
- Тем самым уменьшилась скорость разработки и увеличилась скорость выполнения программного кода.

ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ

- Последнее изменение, происходящее в программировании - переход на ***визуальное программирование***, что предоставляет еще более удобные средства разработки и более быстрое написание кода.
- Лидером в разработке языков визуального программирования является ***Borland***.
- Но визуальное программирование проигрывает ***ООП*** по быстроте работы.
- Несмотря на это лучшая на данный момент технология программирования та, в которой поддерживается ***визуальность***.

ВЫБОР ЯЗЫКА ПРОГРАММИРОВАНИЯ ЗАВИСИТ ОТ РЕШАЕМОЙ ЗАДАЧИ:

- Если нужно написать базы данных, программы общего назначения или утилиты, лучше всего использовать ***Delphi*** или ***C++Builder***.
- Если нужно написать игры, то желательно использовать ***Visual C++*** плюс ***Assembler***.
- Если нужно написать драйверы или программы для работы с «**железом**», то критичен размер файла, и наиболее подходящие языки – чистый ***C*** или ***Assembler***.

ВОПРОСЫ ДЛЯ ОБСУЖДЕНИЯ:

- *Привести классификацию программного обеспечения и проанализировать, какие языки программирования целесообразнее всего использовать для создания различных классов программ.*
- *Проанализировать, чем обоснован выбор языков программирования для написания различных классов программ.*
- *Спрогнозировать перспективы развития языков программирования.*

ТЕМА.
ТЕХНОЛОГИЧЕСКИЙ ЦИКЛ ОБРАБОТКИ ИНФОРМАЦИИ.

ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В ПЕРСОНАЛЬНОМ КОМПЬЮТЕРЕ. ДВОИЧНЫЙ КОД.

- ***Двоичный код*** - это форма записи данных в виде нулей и единиц, т.е. в двоичной системе счисления.
- ***Двоичная система счисления*** это позиционная система счисления с основанием два.
- ***Двоичный код*** применяется во всех цифровых устройствах, т.к. является самым простым и надежным.
- Двоичная арифметика очень простая и ее просто реализовать на аппаратном уровне. Элементы электронных схем, надежнее, если они оперируют терминами «***есть ток***», «***нет тока***».

- Кодируя двумя знаками и используя один двоичный разряд можно закодировать два значения: **0** и **1.**
- Если использовать два двоичных разряда то четыре значения:
  0 0 0 1 1 0 1 1 и т.д.
- В общем случае количество двоичных кодов зависит от разряда чисел, которое вычисляется по формуле: $N = 2^m$,
- где **N** – количество комбинаций нулей и единиц, а
- **m** – количество двоичных разрядов.

- В цифровой техники одному двоичному коду соответствует один логический элемент схемы, который может находиться в двух состояниях - **пропускать ток по схеме** или **нет**.
- Простейшим таким элементом может быть ***двоичный триггер***.

- Вся информация, с которой работает компьютер, кодируется числами.
- Независимо от того, графическая, текстовая или звуковая эта информация, что бы ее мог обрабатывать центральный процессор, она должна тем или иным образом быть представлена числами.

ПРЕДСТАВЛЕНИЕ ТЕКСТОВЫХ ДАННЫХ

- Любой текст состоит из последовательности символов: буквы, цифры, знаки препинания, знаки математических действий, круглые и квадратные скобки и т.д.
- Текстовая информация, как и любая другая, хранится в памяти компьютера в двоичном виде. Для этого каждому символу ставится в соответствие некоторое неотрицательное число, называемое **кодом** символа, и это число записывается в память ЭВМ в двоичном виде.
- Конкретное соответствие между символами и их кодами называется **системой кодировки**.

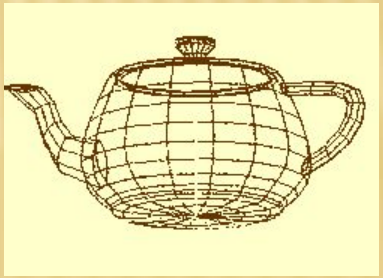
- Для разных типов ЭВМ и операционных систем используются различные таблицы кодировки, отличающиеся порядком размещения символов алфавита в кодовой таблице. Между-народным стандартом на ПК является таблица кодировки **ASCII**.
- Принцип последовательного кодирования алфавита заключается в том, что в кодовой таблице **ASCII** латинские буквы (прописные и строчные) располагаются в алфавитном порядке. Расположение цифр также упорядочено по возрастанию значений.
- Стандартными в этой таблице являются только первые **128** символов, т. е. символы с номерами от **нуля** (двоичный код **00000000**) до **127** (**01111111**). Сюда входят буквы латинского алфавита, цифры, знаки препинания, скобки и некоторые другие символы. Остальные **128** кодов, начиная со **128** (двоичный код **10000000**) и кончая **255** (**11111111**), используются для кодировки букв национальных алфавитов, символов псевдографики и научных символов.

ПРЕДСТАВЛЕНИЕ ИЗОБРАЖЕНИЙ

- Все известные форматы представления изображений можно разделить на **растровые** и **векторные**.
- В **векторном формате** изображение разделяется на примитивы - прямые линии, многоугольники, окружности и сегменты окружностей, параметрические кривые, залитые определенным цветом или шаблоном, и т. д.

- ***Двухмерное векторное изображение***

- ***Трехмерное векторное изображение***



- В растровом формате изображение разбивается на прямоугольную матрицу элементов, называемых ***пикселами***.
- Матрица называется растром. Для каждого пиксела определяется его яркость и, если изображение цветное, цвет.

ПРЕДСТАВЛЕНИЕ ЗВУКОВОЙ ИНФОРМАЦИИ

Существуют два способа звукозаписи:

- **цифровая запись**, когда реальные звуковые волны преобразуются в цифровую информацию путем измерения звука тысячи раз в секунду;
- **MIDI-запись**, которая, вообще говоря, является не реальным звуком, а записью определенных команд-указаний (какие клавиши надо нажимать, например, на синтезаторе). **MIDI**-запись является электронным эквивалентом записи игры на фортепиано.

ПРЕДСТАВЛЕНИЕ ВИДЕО

- Что представляет собой фильм с точки зрения информатики?
- Прежде всего, это **сочетание звуковой и графической информации**.
- Кроме того, для создания на экране эффекта движения используется технология быстрой смены статических картинок.

- Таким образом, рассмотрев принципы хранения в ЭВМ различных видов информации, можно сделать важный вывод о том, что все они так или иначе преобразуются в числовую форму и **кодируются набором нулей и единиц**.
- Благодаря такой универсальности представления данных, если из памяти наудачу извлечь содержимое какой-нибудь ячейки, то принципиально невозможно определить, какая именно информация там закодирована: текст, число или картинка.

ЭТАПЫ РЕШЕНИЯ ЗАДАЧ НА ПЭВМ.

ТЕХНОЛОГИЧЕСКИЙ ЦИКЛ ОБРАБОТКИ ИНФОРМАЦИИ.

ПРОГРАММИРОВАНИЕ —

- это процесс создания (разработки) программы, который может быть представлен последовательностью следующих шагов:

1. ***Спецификация (определение, формулирование требований к программе)*** - один из важнейших этапов, на котором подробно описывается исходная информация, формулируются требования к результату, поведение программы в особых случаях (например, при вводе неверных данных), разрабатываются диалоговые окна, обеспечивающие взаимодействие пользователя и программы.

2. ***Разработка алгоритма*** - необходимо определить последовательность действий, которые надо выполнить для получения результата. Результатом этапа разработки алгоритма является подробное словесное описание алгоритма или его блок-схема.

3. ***Кодирование (запись алгоритма на языке программирования)*** - алгоритм записывается на выбранном языке программирования. В результате получается исходная программа.

4. ***Отладка*** **-** это процесс поиска и устранения ошибок. Ошибки в программе разделяют на две группы: синтаксические (ошибки в тексте) и алгоритмические. Синтаксические ошибки — наиболее легко устраняемые. Алгоритмические ошибки обнаружить труднее. Этап отладки можно считать законченным, если программа правильно работает на одном-двух наборах входных данных.

5. ***Тестирование*** **-** на этом этапе следует проверить, как ведет себя программа на как можно большем количестве входных наборов данных, в том числе и на заведомо неверных.

6. ***Создание справочной системы*** - если разработчик предполагает, что программой будут пользоваться другие, то он обязательно должен создать справочную систему и обеспечить пользователю удобный доступ к справочной информации во время работы с программой.

7. ***Создание установочного диска (CD-ROM)*** - чтобы пользователь мог самостоятельно, без помощи разработчика, установить программу на свой компьютер. Обычно помимо самой программы на установочном диске находятся файлы справочной информации и инструкция по установке программы.