Лабораторная работа №15

Выбор и обоснование выбора среды разработки программы. Изучение различных стилей программирования, правил формирования листинга программы.

Цель работы:

1. Изучить критерии выбора языка программирования. Научиться обосновывать выбор среды разработки в соответствии с критериями выбора языка программирования.

2. Изучить основные парадигмы программирования.

3. Изучить правила формирования листинга программы.

Задание 1

Для реализации статических страниц будут использоваться языки HTML и CSS.

Для реализации интерактивных элементов клиентской части будет использоваться язык JavaScript.

Для реализации динамических страниц должен использоваться язык PHP.

Для работы с БД будет использоваться MySQL

Backend будет написан на C#;

Задание 2

Структурное программирование

В отличие от неструктурного программирования, характеризуется:

- ограниченным использованием условных и безусловных переходов
- широким использованием подпрограмм и прочих управляющих структур (циклов, ветвлений, и т.п.)
- блочной структурой

Языки поддерживающие данную парадигму: Pascal, C#, Java;

Достоинства структурного программирования:

1) повышается надежность программ (благодаря хорошему структурированию при проектировании, программа легко поддается тестированию и не создает проблем при отладке);

2) повышается эффективность программ (структурирование программы позволяет легко находить и корректировать ошибки, а отдельные подпрограммы можно переделывать (модифицировать) независимо от других);

3) уменьшается время и стоимость программной разработки;

4) улучшается читабельность программ.

Главный недостаток:

структурного подхода заключается в следующем: процессы и данные существуют отдельно друг от друга (как в модели деятельности организации, так и в модели программной системы), причем проектирование ведется от процессов к данным. Таким образом, помимо функциональной декомпозиции, существует также структура данных, находящаяся на втором плане.

Пример программы:

Работа с массивом

```
const len = 10000;

begin
  var arr:array[0..len-1] of integer;
  for var i:=0 to len -1 do
  begin
    arr[i]:=random(-100,100);
  end;
  writeln('Первоначальный массив ', arr);

  for var i:=0 to len-2 do //Сортировка массива
  begin
    for var k:=0 to len - 2 - i do
    begin
      if arr[k]>arr[k+1] then swap(arr[k], arr[k+1]);
    end;
  end;
  writeln('Отсортированный массив ',Arr);

  var Find:=readlnInteger('Введите число для поиска ');
  var first:=0;
  var last:=len-1;
  var sred, ind:integer;
  while True do
  begin
   sred:=(last-first) div 2 + First; //нахождение среднего индекса в отсортированном массиве
   if find = arr[sred] then //Если нужное число совпало с числом массива то сохраняем индекс и
выходим из цикла
    begin
      Ind:=sred;
      writeln('Число найдено под индексом ', ind);
      break;
    end
    else
      begin
        if find > arr[sred] then  First:=sred + 1
        else Last:=sred - 1;
```

```
   end;
 end;
end.
```

ООП

Суть ООП заключается в том, чтобы представить программу в виде объектов, которые каким-то образом взаимодействуют друг с другом.

Все, что угодно, можно представить в виде объекта: человека, воздушный шарик, сообщение в мессенджере. У объекта могут быть свойства, например, цвет – красный, размер – большой. Также у объекта могут быть методы для совершения операций. Например, если объект телевизор, вызываем метод «включить», и телевизор включается.

Объект — это экземпляр какого-то класса. Класс — это шаблон, в котором описаны все свойства будущего объекта и его методы. При этом если класс воздушного шарика определяет свойство цвет, то сам класс никакого значения цвета не имеет. Но экземпляры этого класса, которых, к слову, можно создавать сколько угодно, уже будут раскрашены в любые цвета.

Языки поддерживающие данную парадигму: Pascal, C#, Python;

Преимущества ООП:

- Возможность легкой модификации (при грамотном анализе и проектировании)
- Возможность отката при наличии версий
- Более легкая расширяемость
- «Более естественная» декомпозиция программного обеспечения, которая существенно облегчает его разработку.
- Сокращение количества межмодульных вызовов и уменьшение объемов информации, передаваемой между модулями.
- Увеличивается показатель повторного использования кода.

Недостатки ООП:

- Требуется другая квалификация

- Резко увеличивается время на анализ и проектирование систем
- Увеличение времени выполнения
- Размер кода увеличивается :interrobang:
- Неэффективно с точки зрения памяти (мертвый код - тот, который не используется) :interrobang:
- Сложность распределения работ на начальном этапе
- Себестоимость больше

Пример программы:

Программа выдачи зарплаты сотрудникам в python

```
class Person:
    def __init__(self, name, j = 'безработный', pay=0):
        self.name = name
        self.__job = j
        self.pay = pay
    def __str__(self):
        return self.name+', работа - '+self.__job+', зарплата -
'+str(self.pay)+'$'
    @property
    def family(self):
        return str(self.name.split()[:1])
    def upPay(self, percent = 0):
        self.pay = int(self.pay + (percent/100 * self.pay))
    def downPay(self, percent = 0):
        self.pay = int(self.pay - (percent/100 * self.pay))
    @property
    def job(self, job):
        return self.__job
    @job.setter
    def job(self, job):
        self.__job = job
        print('Теперь '+self.name+' работает '+self.__job+'ом')
    @job.deleter
    def job(self):
        self.__job = 'безработный'
        print('Теперь '+self.name+' безработный ')

class Manager(Person):
    def __init__(self, name, j = 'безработный', pay=0, category='1'):
        Person.__init__(self,name, j, pay)
        self.category = category
    def __str__(self):
        return Person.__str__(self)+', категория - '+str(self.category)
    def upPay(self, percent = 0, bonus = 10):
        Person.upPay(percent+bonus)

Alexey = Person('Дунников Алексей', 'фронтенд разработчик', 9700)
Pavel = Person('Шишко Павел')
Albert = Manager('Войтюль Альберт', 'менеджер по продажам', 1200, 2)
```

```
if __name__ == '__main__':
    print(Alexey)
    print(Pavel)
    print(Alexey.family)
    Alexey.upPay(30)
    print(Alexey)
    Alexey.job = 'сварщик'
    print(Alexey)
    del Alexey.job
    print(Alexey)

    print('---------Менеджер----------')
    print(Albert)
    Albert.downPay(20)
    print(Albert)
    del Albert.job
    print(Albert)
```

Я буду использовать модульное программирование, так как: Модульное программирование — это способ создания программы посредством объединения модулей в единую структуру. Применение способа позволяет значительно повысить скорость разработки, обеспечить ее надежность, упростить тестирование.

Так же для сайта буду использовать Объектноориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования. Что очень сильно поможет в разработке сайта.

Задание 3

Листинги разработанных программ должны располагаться в отдельных приложениях с обязательными ссылками на них.

Программный код должен быть сопровожден комментариями. Рекомендуется использовать возможности самодокументирования кода.

В основной части работы для иллюстрации излагаемого теоретического материала должны приводиться листинги фрагментов программ, которые следует располагать непосредственно после текста, в котором они впервые упоминаются. На все листинги должны быть даны ссылки в тексте работы.

При оформлении листингов следует использовать шрифт Courier New, размер – 12 пт, межстрочный интервал – одинарный. Рекомендуется отделять смысловые блоки пустыми строками, а также визуально обозначать вложенные конструкции с помощью отступов.

Ключевые слова и комментарии рекомендуется выделять с помощью различных начертаний шрифта. Таким же образом в основном тексте работы должны обозначаться имена библиотек, подпрограмм, констант, переменных, структур данных, классов, их поля и методы.
Листинги должны иметь порядковую нумерацию в пределах каждого раздела. Номер листинга должен состоять из номера раздела и порядкового номера листинга, разделенных точкой, например: «Листинг 3.2» – второй листинг третьего раздела. Если в работе содержится только один листинг, он обозначается «Листинг 1». При ссылке на листинг следует писать слово «листинг» с указанием его номера.

Название листинга печатается тем же шрифтом, что и основной текст, и размещается над листингом слева, без абзацного отступа через тире после номера листинга.