Текстовый материал для ответа на вопросы госэкзамена по теме

Объектно-ориентированное программирование.

Объектно-ориентированный подход в программировании. Понятие класса и объекта. Поля, методы и свойства объектов. Создание и удаление объектов. Описание пользовательских конструкторов.

Объектно-ориентированное программирование (ООП) — это методика разработки программ, в основе которой лежит понятие объекта, как некоторой структуры, описывающей объект реального мира, его поведение. Задача, решаемая с помощью методики ООП, описывается в терминах объектов и операций над ними, а программа при таком подходе представляет собой набор объектов и связей между ними.

Расширяя рамки сложного типа данных — записи (record), концепция объектно-ориентированного программирования позволяет программисту определять **классы**. Тип данных **класс** – сложная структура, включающая в себя помимо описания данных описание процедур и функций, которые могут быть выполнены над представителем класса — **объектом.**

Пример описания простого класса:

```
type
      TPerson = class
        private
          fname     : string[15];
          faddress: string[40];
        public
          procedure Show;
      end;
```

Данные класса называются **полями**, процедуры и функции – **методами**[1]. В приведенном примере `TPerson` – это имя класса, `fname` и `faddress` – имена полей, а `Show` – имя метода[2]. С**войства** - виртуальные характеристики объектов, базирующиеся на значениях конкретных полей. Оспределение значения свойства и изменение его значения выполняются в рамках соответствующих методов[3].

Объекты — экземпляры класса описываются в разделе `var`:

```
var
      Student  : TPerson;
      Professor  : TPerson;
```

Переменная типа объект является ссылочной. Она хранит адрес динамической структуры[4] в памяти, содержащей ссылку на класс объекта и значения его полей. В отличие от полей, которые у каждого объекта свои, методы у разных объектов одного класса общие (поэтому нет необходимости каждому объекту хранить адреса его методов – достаточно ссылки на класс). От

---

[1] Если нет специального уточнения при описании (как в данном примере), то по способу вызова методы относятся к группе статических. Кроме того, методы могут быть виртуальными и динамическими (см. раздел "Виртуальные и динамические методы").

[2] Назначение секций `private` и `public` в описании класса см. в разделе "Области видимости".

[3] Подробнее см. раздел «Инкапсуляция».

[4] Пример распределения памяти для объектов и классов см. в разделе "Структура объекта".

обычных процедур и функций методы отличаются тем, что при вызове им передается указатель на тот объект, который вызвал метод. Внутри метода этот объект доступен под зарезервированным именем `Self`.

Создание и удаление объектов.

Так как экземпляры объектов являются динамическими, то для работы с ними предварительно требуется произвести необходимые действия по их созданию в куче. Эти действия выполняют специальные методы - **конструкторы** объектов. Любой конструктор перед выполнением записанных в его теле операторов вначале резервирует в куче место, необходимое для размещения объекта, и заполняет поля созданного объекта нулевыми значениями. После выполнения содержащихся в его теле операторов, конструктор, являясь функцией, возвращает адрес нового объекта. По умолчанию в каждом классе доступен стандартный конструктор `Create`, не содержащий внутренних операторов.

Созданный экземпляр объекта уничтожается другим методом — **деструктором**. Для разрушения объектов можно использовать деструктор `Destroy`, доступный любому объекту по умолчанию. Однако чаще всего применяют стандартный метод `Free`. Пользователь вправе написать свои конструктор или деструктор, указав их в описании класса.

Жизненный цикл объекта, состоящий из трех этапов (создание, использование и разрушение), может выглядеть примерно так:

```
Student:=TPerson.Create; // создание объекта
...
Writeln(Student.fname)   // использование объекта
...
Student.Free;            // разрушение объекта
```

Обратите внимание, имена вызываемых методов и полей объекта записываются через точку после имени объекта. Однако для метода `Create` сделано исключение: его применение к объекту напрямую невозможно, т.к. самого объекта еще не существует (т.н. ошибка доступа). Поэтому функция `Create` вызывается как метод класса (в данном случае `TPerson`) и возвращаемый ею адрес объекта присваивается ссылке `Student`.

Ниже представленный класс описывает точку на плоскости с координатами (x,y). Переменная данного класса содержит два поля (x и y), три метода (Ro, Move, GetSection) конструктор CreateLine и свойство Section (только для чтения).

```
type
      TPoint = class
        x,y   : Real;
        constructor CreateLine;
        function Ro : real;
        proceure Move(Dx,Dy : real);
        function GetSection : byte;
        property Section: byte read GetSection;
      end;

constructor TPoint.CreateLine; // создает точку с координатами (1,1)
begin
      x:=1;
      y:=1
end;
```

```
function TPoint.Ro; // вычисляет расстояние до начала координат
begin
      result:=Sqrt(Sqr(x)+Sqr(y));
end;

procedure TPoint.Move; // перемещает точку на вектор (Dx,Dy)
begin
      x:=x+Dx;
      y:=y+Dy;
end;

function TPoint.GetSection;
begin
      if x*y=0
        then Result:=0
        else if x >0
              then if y>0 then result:=1
                          else result:=4
              else if y>0 then result:=2
                          else result:=3
end;
```

Жизненный цикл объекта данного класса может выглядеть примерно так:

```
var   a  : Tpoint;
begin
      a:=TPoint.Create;
      ...
      a.Move(5,-1.6);
      ...
      a.Free;
end
```