**При исправлении ошибок кода иметь в виду:**

* Поля данных должны объявляться перед методами
* Имена полей данных объекта не должны совпадать с именами формальных параметров у методов объекта
* Если в описании объекта есть виртуальные методы, то обязательно должна быть специальная процедура Constructor

**Полиморфизм** позволяет по-разному использовать методы с одинаковыми именами в иерархии типов объектов (виды наследования).

**? Приведите пример простого переопределения методов и объясните, как будут работать эти методы**

type

t1=object

a:integer;

procedure p1;

function f1(a1:string):string;

end;

t2=object(t1)

function f1:string;

end;

*Function t1.f1;*

*Begin*

*f1:=a1 ;*

*end;*

Procedure t1.p1;

Begin

writeln(f1('==='))

end;

*Function t2.f1:string;*

*Begin*

*f1:='function t2.f1'*

*end;*

var

o1:t1;o2:t2;

Begin

...

o1.p1;

o2.p1;

...

end.

В результате работы программы будут напечатаны одинаковые 2 строки ("==="), так как вызов функции f1 хоть и осуществляется в теле процедуры p1, вызванной в объекте o2, но процедура описана в типе t1 (предок), поэтому будет вызвана функция f1, описанная в этом же типе предка.

**? Что происходит при выполнении операторов присваивания, когда справа и слева стоят объекты. Каким условиям должны удовлетворять эти объекты**

Данные объекта с правой стороны равенства будут скопированы в данные объекта с левой стороны равенства. Если – предок := потомок, то лишние поля игнорируются. Но нельзя присваивать предка потомку.

**? Как инициализировать статический объект с виртуальными методами**

Если в описании типа объекта есть виртуальные методы, то обязательно должна быть специальная процедура, называемая Constructor. Перед запуском виртуального метода объекта должен быть запущен Constructor, только после ее выполнения будет определен путь к виртуальному методу объекта.

**? Внешняя процедура (функция) для инициализации полей объекта типа t1 и вызов ее в головной программе**

x (var o : t1; fa : integer; fb : real); – формальные параметры процедуры (функции), сначала по ссылке передаем объект, а затем по значению требующие инициализации поля.

Тело процедуры

with o do begin

a := fa;

b := fb;

end;

Вызов в г.п.

x(o1, 2, 1.5);

**? Что такое позднее связывание метода с объектом и на каком этапе обработки программы оно осуществляется**

Позднее связывание - связь виртуального метода с объектом, осуществляемая на этапе исполнения программы при запуске процедуры Constructor, после чего появляется возможность использовать этот виртуальный метод.

**? В какой области ОП и в каком порядке хранятся коды методов типов объектов**

Коды всех методов хранятся в порядке их описания в сегменте кода. Дополнительно выделяется место в поле каждого экземпляра объектов для дальнейшего размещения смещения таблицы виртуальных методов этого типа объекта.

**? Как обратиться к методу предка и объекта- потомка в случае переопределения методов**

К методу предка из объекта-потомка можно обратиться при помощи слова inherited. Например, inherited f, это значит, что программа должна обратиться не к переопределенному методу f в текущем типе, а к методу f, описанному в ближайшем предке типа.

**? Написать процедуру инициализации полей a1 и a2 у объектов o1 и o2**

procedure t1.p1 (var a5 : integer; c : real);

Begin

a1 := a5;

a2 := c;

end;

В г.п.

o1.p1 (10, 10.5);

o2.p1 (25, 25.5);

**? На каком этапе обработки программы процессор выделяет ОП для объектов и каков объем этих объектов (в байтах)**

Процессор выделяет ОП для объектов на этапе компиляции. Сначала выделяется место для полей данных всех объектов в сегменте данных и место для всех методов объектов в сегменте кода, происходит раннее связывание. Если есть виртуальные методы, то строятся таблицы виртуальных методов, в которые затем на этапе исполнения программы при запуске конструкторов (позднее связывание) будут внесены точки входа этих виртуальных методов, а сами таблицы станут связаны по адресу с полем экземпляра объекта (сегментом данных).

|  |
| --- |
| **Сегмент данных** |
| Поля объектов и их смещение |
| Адресс ТВМ (2 байта) |

|  |  |
| --- | --- |
| **Сегмент кода** | |
| Смещение | Метод |
| Коды методов в порядке их описания | |

**? Что такое точки входа в виртуальный метод и где они хранятся**

Точка входа — адрес в оперативной памяти, с которого начинается выполнение программы. Точки входа в виртуальные методы появляются после соответствующей процедуры Constructor. Точки входа в виртуальные методы объектов данного типа хранятся в таблицах виртуальных методов.

**? Для чего нужен деструктор**

Диструктор нужен для особождения памяти от динамического объекта (при использовании расширенного синтаксиса процедуры Dispose). Диструктор использует таблицу виртуальных методов, в которой хранится размер динамического объекта, и поэтому освобождение памяти происходит правильно

**? Что такое инкапсуляция**

Инкапсуляция - объединение полей данных и действий над ними (методов) в описании объектного типа.

**? Для чего нудны виртуальные методы**

Методы описываются как виртуальные, если у предка и потомка имена методов и списки параметоров совпадают, а тела методов различны, и имеется необходимость потомку использовать свой метод.

**? Как проинициализировать динамический объект с виртуальными методами**

Для инициализации динамического объекта с виртуальными методоми нужно использовать расширенный синтаксис процедуры New: new(указатель на объект, имя его конструктора). Например, new (u, init);

|  |  |
| --- | --- |
| **При исправлении ошибок кода иметь в виду:**   * Поля данных должны объявляться перед методами * Имена полей данных объекта не должны совпадать с именами формальных параметров у методов объекта * Если в описании объекта есть виртуальные методы, то обязательно должна быть специальная процедура Constructor   **? Приведите пример простого переопределения методов и объясните, как будут работать эти методы**  type  t1=object  a:integer;  procedure p1;  function f1(a1:string):string;  end;    t2=object(t1)  function f1:string;  end;    *Function t1.f1;*  *Begin*  *f1:=a1 ;*  *end;*    Procedure t1.p1;  Begin  writeln(f1('==='))  end;    *Function t2.f1:string;*  *Begin*  *f1:='function t2.f1'*  *end;*  var  o1:t1;o2:t2;  Begin  ...  o1.p1;  o2.p1;  ...  end.  В результате работы программы будут напечатаны одинаковые 2 строки ("==="), так как вызов функции f1 хоть и осуществляется в теле процедуры p1, вызванной в объекте o2, но процедура описана в типе t1 (предок), поэтому будет вызвана функция f1, описанная в этом же типе предка.  **? Что происходит при выполнении операторов присваивания, когда справа и слева стоят объекты. Каким условиям должны удовлетворять эти объекты**  Данные объекта с правой стороны равенства будут скопированы в данные объекта с левой стороны равенства. Если – предок := потомок, то лишние поля игнорируются. Но нельзя присваивать предка потомку.  **? Как инициализировать статический объект с виртуальными методами**  Если в описании типа объекта есть виртуальные методы, то обязательно должна быть специальная процедура, называемая Constructor. Перед запуском виртуального метода объекта должен быть запущен Constructor, только после ее выполнения будет определен путь к виртуальному методу объекта. | |
| **При исправлении ошибок кода иметь в виду:**   * Поля данных должны объявляться перед методами * Имена полей данных объекта не должны совпадать с именами формальных параметров у методов объекта * Если в описании объекта есть виртуальные методы, то обязательно должна быть специальная процедура Constructor   **? Внешняя процедура (функция) для инициализации полей объекта типа t1 и вызов ее в головной программе**  x (var o : t1; fa : integer; fb : real); – формальные параметры процедуры (функции), сначала по ссылке передаем объект, а затем по значению требующие инициализации поля.  Тело процедуры  with o do begin  a := fa;  b := fb;  end;  Вызов в г.п.  x(o1, 2, 1.5);  **? Что такое позднее связывание метода с объектом и на каком этапе обработки программы оно осуществляется**  Позднее связывание - связь виртуального метода с объектом, осуществляемая на этапе исполнения программы при запуске процедуры Constructor, после чего появляется возможность использовать этот виртуальный метод.  **? В какой области ОП и в каком порядке хранятся коды методов типов объектов**  Коды всех методов хранятся в порядке их описания в сегменте кода. Дополнительно выделяется место в поле каждого экземпляра объектов для дальнейшего размещения смещения таблицы виртуальных методов этого типа объекта.  **? Как обратиться к методу предка и объекта- потомка в случае переопределения методов**  К методу предка из объекта-потомка можно обратиться при помощи слова inherited. Например, inherited f, это значит, что программа должна обратиться не к переопределенному методу f в текущем типе, а к методу f, описанному в ближайшем предке типа. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **При исправлении ошибок кода иметь в виду:**   * Поля данных должны объявляться перед методами * Имена полей данных объекта не должны совпадать с именами формальных параметров у методов объекта * Если в описании объекта есть виртуальные методы, то обязательно должна быть специальная процедура Constructor   **? Написать процедуру инициализации полей a1 и a2 у объектов o1 и o2**  procedure t1.p1 (var a5 : integer; c : real);  Begin  a1 := a5;  a2 := c;  end;  В г.п.  o1.p1 (10, 10.5);  o2.p1 (25, 25.5);  **? На каком этапе обработки программы процессор выделяет ОП для объектов и каков объем этих объектов (в байтах)**  Процессор выделяет ОП для объектов на этапе компиляции. Сначала выделяется место для полей данных всех объектов в сегменте данных и место для всех методов объектов в сегменте кода, происходит раннее связывание. Если есть виртуальные методы, то строятся таблицы виртуальных методов, в которые затем на этапе исполнения программы при запуске конструкторов (позднее связывание) будут внесены точки входа этих виртуальных методов, а сами таблицы станут связаны по адресу с полем экземпляра объекта (сегментом данных).   |  |  | | --- | --- | | **Сегмент данных** | | | Поля объектов и их смещение | | | Адресс ТВМ (2 байта) | | | **Сегмент кода** | | | Смещение | Метод | | Коды методов в порядке их описания | |   **? Что такое точки входа в виртуальный метод и где они хранятся**  Точка входа — адрес в оперативной памяти, с которого начинается выполнение программы. Точки входа в виртуальные методы появляются после соответствующей процедуры Constructor. Точки входа в виртуальные методы объектов данного типа хранятся в таблицах виртуальных методов.  **? Для чего нужен деструктор**  Диструктор нужен для особождения памяти от динамического объекта (при использовании расширенного синтаксиса процедуры Dispose). Диструктор использует таблицу виртуальных методов, в которой хранится размер динамического объекта, и поэтому освобождение памяти происходит правильно |

|  |
| --- |
| **При исправлении ошибок кода иметь в виду:**   * Поля данных должны объявляться перед методами * Имена полей данных объекта не должны совпадать с именами формальных параметров у методов объекта * Если в описании объекта есть виртуальные методы, то обязательно должна быть специальная процедура Constructor   **? Что такое инкапсуляция**  Инкапсуляция - объединение полей данных и действий над ними (методов) в описании объектного типа.  **? Для чего нудны виртуальные методы**  Методы описываются как виртуальные, если у предка и потомка имена методов и списки параметоров совпадают, а тела методов различны, и имеется необходимость потомку использовать свой метод.  **? Как проинициализировать динамический объект с виртуальными методами**  Для инициализации динамического объекта с виртуальными методоми нужно использовать расширенный синтаксис процедуры New: new(указатель на объект, имя его конструктора). Например, new (u, init); |