**Московский государственный
университет им. М.В. Ломоносова**

Факультет вычислительной математики и кибернетики

И.Г. Головин

Курс «Языки программирования»

Варианты письменного экзамена

Методическое пособие

Москва

2009

УДК 519.68
ББК

Печатается по решению Редакционно-издательского совета факультета вычислительной математики и кибернетики МГУ им. М. В. Ломоносова

Рецензенты:
проф., д.ф.-м.н. Машечкин И. В.
доцент, к.ф.-м.н. Терехин А.Н.

И.Г.Головин

Курс «Языки программирования». Варианты письменного экзамена: Методическое пособие для студентов III курса. - М.: Издательский отдел факультета ВМиК МГУ им. М.В.Ломоносова (лицензия ИД № 05899 от 24.09.2001), 2009 – 32 с.

ISBN 978-5-89407-407-8

Методическое пособие содержит варианты письменных экзаменов по курсу лекций «Языки программирования», читаемого для студентов 3 курса 3 потока факультета ВМК МГУ. Цель письменного экзамена по курсу – на основе унифицированного набора задач по различным разделам курса и общих критериев объективно оценить знания студентов, уровень их подготовки. Варианты включают задачи различного уровня сложности: наряду с простейшими задачами и вопросами по конкретной теме, имеются задачи, предполагающие знание материала нескольких тем и умение владеть этим материалом. В пособии приведены ответы, а также продемонстрированы авторские решения некоторых задач, дана программа курса и рекомендуемая литература.

Пособие рекомендуется студентам при подготовке к письменному экзамену.

УДК 519.68

ББК

ISBN 978-5-89407-407-8

I. Программа курса «Языки программирования»

1. Введение

Определение языков программирования (ЯП), ЯП и основные парадигмы программирования. Исторический очерк развития ЯП. Основные позиции при рассмотрении ЯП. Схема рассмотрения ЯП: базис, средства развития и средства защиты.

Основные понятия языков программирования: данные, операции и связывание.

Понятие о виртуальной машине языка.

2. Базисные типы данных в языках программирования: простые и составные типы данных, операции над ними

Простые типы данных.

Арифметические типы данных: целые, плавающие, фиксированные. Проблемы представления чисел и способы их решения в ЯП.

Символьные и логические типы данных.

Порядковые типы: диапазоны и перечисления. Особенности реализации перечислений в современных ЯП.

Ссылки и указатели. Управление памятью. Автоматическая сборка мусора. Объектно-референциальная модель в современных ЯП.

Составные типы данных.

Массивы и их особенности в современных ЯП.

Записи. Недостатки системы типов в традиционных ЯП. Объединения как средство преодоления этих недостатков. Проблемы, связанные с объединениями.

Ассоциативные массивы и записи.

3. Операторный базис языков программирования. Управление последовательностью вычислений

Операторы присваивания.

Понятие о структурном программировании. Разновидности управляющих конструкций в современных языках программирования.

Условные операторы и многовариантные развилки. Циклы. Оператор перехода, связанные с им проблемы и способы их решения в современных ЯП.

Особенности реализации циклов-итераторов в современных ЯП.

4. Процедурные абстракции

Подпрограммы и сопрограммы. Операторы возврата и возобновления.

Процедуры и функции в современных ЯП.

Передача параметров: семантика и способы реализации.

Статический полиморфизм и перегрузка имен подпрограмм.

Подпрограммные типы данных. Проблемы, связанные с подпрограммными типами данных и способы их решения в современных ЯП.

5. Определение новых типов данных. Логические модули. Классы.

Концепция уникальности типа в традиционных языках программирования и понятие строгой типизации.

Конструкции объявления новых типов данных. Тип данных как множество значений и множество операций. Походы к определению новых типов данных: модули и классы.

Понятие логического модуля. Использование модулей для определения новых типов данных. Особенности понятия модуля в современных ЯП. Импорт и экспорт имен. Видимость имен: непосредственная и потенциальная. Управление видимостью. Области видимости и пространства имен.

Модульность и технология программирования: проектирование «сверху-вниз» и «снизу-вверх».

Понятие класса. Класс как тип данных. Члены класса: функции, данные. Статические и нестатические члены. Члены - вложенные классы. Статические и нестатические классы. Классы и области видимости.

Понятие специальных функций-членов. Проблема инициализации объектов и способы ее решения. Конструкторы, деструкторы, операторы `using` и `try-finally`.

Преобразование типов и классы. Явные и неявные преобразования. Управление преобразованиями в современных ЯП: проблемы и способы их решения.

Классы и перегрузка имен. Перегрузка встроенных знаков операций.

Итераторы и индексаторы.

Классы и стандартные библиотеки. Встроенные классы стандартной библиотеки.

6. Инкапсуляция и абстрактные типы данных.

Понятие инкапсуляции. Единицы и атомы защиты. Понятие абстрактного типа данных (АТД) и его достоинства.

Инкапсуляция и логические модули. Управление видимостью. Реализация АТД в модульных языках программирования (Ада, Оберон, Модула).

Инкапсуляция и классы. Управление видимостью и управление доступом. Пространства имен и инкапсуляция. Реализация АТД с помощью понятия класса.

Принцип разделения определения, реализации и использования (РОРИ). Эволюция принципа РОРИ в современных ЯП.

7. Модульность и раздельная трансляция

Виды трансляции. Физические модули. Программная и трансляционная библиотеки. Раздельная трансляция: зависимая и независимая. Недостатки независимой трансляции и способы их преодоления.

Особенности зависимой трансляции в современных ЯП. Одностороняя и двустороняя связь модулей и раздельная трансляция.

Раздельная трансляция и пространства имен.

8. Исключительные ситуации и обработка ошибок

Понятие исключительной ситуации (ИС) и его эволюция. ИС и ошибки в программах. Четыре аспекта рассмотрения ИС: определение, возникновение, распространение и обработка. Воплощение этих аспектов в современных ЯП.

Два подхода к обработке ИС: семантика возобновления и семантика завершения. Их сравнение. Семантика завершения и современные ЯП. Свертка стека. Оператор `try-finally`.

Дополнительные особенности ИС: спецификация ИС, проверяемые и непроверяемые ИС.

9. Наследование типов и классов

Концепция уникальности типов в традиционных языках и строгая типизация в объектно-ориентированных языках. Понятие единичного наследования. Единичное наследование в современных ЯП. Наследование и модель представления объекта в памяти. Преобразование из производного типа в базовый. Иерархии типов, статические и динамические типы в объектно-ориентированных ЯП.

Наследование и области видимости имен. Замещение, перегрузка и скрытие имен при наследовании. Наследование и инкапсуляция. Управление видимостью и доступом при наследовании.

Запрещение наследования для классов и методов.

Наследование и специальные функции. Понятие о множественном наследовании.

10. Динамический полиморфизм

Статическое и динамическое связывание методов. Динамический тип данных и динамическое связывание. Замещение функций и динамическое связывание. Особенности динамического связывания в современных ЯП. Достоинства и недостатки динамического связывания. Снятие динамического связывания. Механизм реализации динамического связывания на примере языка Си++. Таблица виртуальных методов.

Понятие о мультиметодах.

11. Абстрактные классы и интерфейсы

Понятие абстрактного класса (АК). Необходимость понятия АК при проектировании иерархий классов. Воплощение концепции АК в современных ЯП.

Абстрактные классы и интерфейсы. Интерфейс как языковая конструкция. Связь интерфейсов и других языковых конструкций (итераторов, сохраняемых объектов и т.д.). Интерфейсы и иерархии классов. Множественное наследование интерфейсов. Реализация интерфейсов и ее особенности современных ЯП. Явная и неявная реализация интерфейсов.

12. Множественное наследование

Множественное наследование в языке Си++. Ромбовидное наследование и его примеры. Проблемы множественного наследования: конфликт имен, реализация динамического связывания. Особенности реализации множественного наследования при наследовании интерфейсов.

13. Динамическая идентификация типа

Понятие о динамической идентификации типа (ДИТ). Достоинства и недостатки использования ДИТ. Особенности ДИТ в современных языках программирования.

14. Понятие о родовых объектах. Обобщенное программирование

Понятие о статической параметризации и родовых объектах. Достоинства статической параметризации. Статическая параметризация и ООП.

Родовые модули и подпрограммы в языке Ада.

Механизм шаблонов в языке Си++. Шаблоны-классы и шаблоны-функции. Параметры шаблонов. Вывод параметров шаблонов. Генерация кода по шаблонам. Частичная специализация шаблонов. Обобщенное программирование на языке Си++: функторы, свойства, стратегии, шаблоны выражений.

Сравнение механизма шаблонов Си++ и родовых объектов Ады.

Особенности родовых объектов в языках C# и Java.

II. Варианты экзаменационных работ

2.1. Вариант 2003

1. Что будет напечатано в результате работы следующей программы на языке Си++?

```
#include <iostream.h>
class A {
public:
    virtual void f() {cout << "A::f\n"; g();}
    virtual void g() { cout << "A::g\n"; }
    void h() { cout << "A::h\n"; }
};
class B : public A {
public:
    void f() { cout << "B::f\n";  }
    void g() { cout << "B::g\n"; h(); }
    void h() { cout << "B::h\n"; }
};
class C : public B {
public:
    void f() { cout << "C::f\n"; }
    void g() { cout << "C::g\n"; }
    void h() { cout << "C::h\n"; f(); }
};
void P(A*pa,B*pb) {
    pa->f(); pa->g(); pa->h();
    pb->f(); pb->g(); pb->h();
    delete pa; delete pb;
}
int main() {
    P(new B, new B);
    cout<<"------------------------------\n";
    P(new C, new C); return 0;
}
```

2. В каких из перечисленных ниже языков есть оператор перехода «goto метка»?

Ада 83, Ада 95, Оберон, Оберон-2, Модула-2, Java, Delphi, C#

3. Сравните между собой конструкции "uses" в языке Delphi и "use" в языке Ада (для чего применяются, сходства, отличия).

4. В каких из перечисленных ниже языков длина массива является только статическим атрибутом? Приведите пример массива с динамическим атрибутом - длиной для какого-либо языка.

Ада, Си++, Оберон, Модула-2, Java, C#

5. Что означают термины "семантика возобновления" и "семантика завершения" при обработке исключений? Для каждого способа (семантики) приведите пример языка, где этот способ (семантика) используется.

6. Назовите хотя бы один язык, в котором нельзя передавать подпрограммы как параметры других подпрограмм.

7. В каких классах памяти могут размещаться данные в языках программирования? В каких классах памяти размещаются объекты классов языка C#?

8. Ниже приведена спецификация родового пакета Stacks на языке Ада. Напишите объявление шаблонного класса на языке Си++, предназначенного для той же роли, что и этот пакет.

```
GENERIC
     TYPE T IS PRIVATE; SIZE : INTEGER;
PACKAGE Stacks IS
     TYPE Stack IS LIMITED PRIVATE;
     PROCEDURE  Push(S: IN OUT Stack; X : IN T);
     PROCEDURE  Pop(S: IN OUT Stack; X : OUT T);
     FUNCTION IsEmpty(S : IN Stack) RETURN BOOLEAN;
     FUNCTION IsFull(S : IN Stack) RETURN BOOLEAN;
PRIVATE
     TYPE Stack is RECORD
          Body : ARRAY (1..SIZE) OF T;
          Top : INTEGER := 1;
     END RECORD;
END Stacks;
```

2.2. Вариант 2004

1. Что будет напечатано в результате работы следующей программы на Си++?

```
#include <iostream.h>
class X {
public:
    virtual void f() {cout << "X::f\n"; g(); }
    void g() { cout << "X::g\n";}
};
class Y : public X {
public:
    void f() { cout << "Y::f\n";  }
    void g() { cout << "Y::g\n"; f();}
};
class Z : public Y {
public:
    void f() { cout << "Z::f\n"; }
    void g() { cout << "Z::g\n"; f();}
};
void P(X*px,Y*py) {
    px->f(); px->g();
    py->f(); py->g();
    delete px; delete py;
}
int main() {
    P(new X, new Y);
    cout<<"----------------------------\n";
    P(new Y, new Z); return 0;
}
```

2. В каких из перечисленных ниже языков есть понятие динамического связывания подпрограмм (методов класса)?

Ада 83, Ада 95, Си++, Оберон, Оберон-2, Модула-2, Java, Delphi, C#

3. Напишите на языке Ада 95 объявления, эквивалентные приведенным ниже описаниям на языке Оберон-2.

```
TYPE T* = RECORD I*, J : INTEGER; END;
TYPE T1* = RECORD (T) K : INTEGER; END;
PROCEDURE (VAR X: T) P* (L : INTEGER);
PROCEDURE (VAR X: T1) P* (L : INTEGER);
```

4. В каких из перечисленных ниже языков есть конструкция try-finally? Объясните её смысл для какого-нибудь языка.

Ада, Си++, Оберон, Модула-2, Java, Delphi, C#

5. Сколько конструкторов имеет класс S, описанный на языке Си++? Ответ обоснуйте.

```
struct S { explicit S(int); double i,j; };
```

6. Объясните смысл конструкции package в языке Java.

7. Смоделируйте на языке Оберон понятие, аналогичное скрытому типу данных на языке Модула-2.

8. Смоделируйте на языке Си++ функцию

```
void f() throw (E1,E2,E3) { g(); h(); }
```

предполагая, что конструкция throw не допускается компилятором.

2.3. Вариант 2005

1. Что будет напечатано в результате работы следующей программы на Си++?

```
#include <iostream>
using namespace std;
class A {
public:
    virtual void f() {cout << "A::f\n"; g(); }
    void g() { cout << "A::g\n";}
};
class B : public A {
public:
    void f() { cout << "B::f\n";  }
    void g() { cout << "B::g\n"; f();}
};
class C : public B {
public:
    void f() { cout << "C::f\n"; }
    void g() { cout << "C::g\n"; f();}
};
void P(A*pa,B& b) {
    pa->f(); pa->g();
    b.f(); b.g();
    delete pa;
}
int main() {
     B b;
    P(new A, b);
    cout<<"------------------------------\n";
     C c;
    P(new B, c); return 0;
}
```

2. Объясните, что означает термин «семантика возобновления» при обработке исключительных ситуаций. Приведите пример моделирования семантики возобновления на языке Си++.

3. Напишите спецификацию абстрактного типа данных Deque (очередь с двумя “хвостами”) на языках Ада и Java (тела методов и тело пакета можно опустить).

4. Объясните, что означает термин «абстрактная функция». В каких из перечисленных ниже языков есть соответствующее понятие?

Ада 83, Ада 95, Си++, Модула-2, Java, Delphi, C#

5. Что означает ключевое слово super на языке Java? Есть ли его аналог в языке C#? Если есть, то приведите пример на каждом из этих языков.

6. В каких из перечисленных ниже языков есть конструкция «свойство» (property)? Объясните, что она означает (на примере какого-либо языка).

Java, Ада 83, Ада 95, Си++, Delphi, Оберон, Оберон-2, Модула-2, C#

7. Объясните, что означает термин «перегрузка»(overloading). В каких из перечисленных ниже языков есть соответствующее понятие?

Ада 83, Ада 95, Си++, Модула-2, Java, Delphi, C#, Оберон, Оберон-2

8. Чем отличается деструктор языка Си++ от деструктора языка C#?

2.4. Вариант 2006

1. В каких языках из перечисленных ниже есть понятие «размеченное объединение»? Объясните, что оно означает.

Ада, Си++, Оберон, Модула-2, Java, C#, Паскаль

2. Назовите две причины большей надежности указателей языка Ада 83 по сравнению с указателями языка Си++.

3. Объясните, что означает понятие «раздельная независимая трансляция». В каких языках из перечисленных ниже есть это понятие?

Ада, Си, Си++, Оберон, Оберон-2, Модула-2, Java, C#, Паскаль

4. Объясните смысл конструкции **where** в языке C# (версия 2.0).

5. Что будет напечатано в результате работы следующей программы на Си++?

```
#include <iostream>
using namespace std;
class X {
public:
   X() { f(); cout << '\n';}
   virtual void g(){cout<<1<<' ';}
   void f() { g();}
};
```

```
class Y: public X {
public:
   Y() {f(); cout << '\n';}
   void g() { cout<< 2<< ' ';}
   void f() { g(); }
};
class Z: public Y {
public:
   Z(){g(); f(); cout << '\n';}
   void g() { cout << 3 << ' '; }
   void f() { g(); }
};

X x; Y y; Z z;
X*px = &x; Y*py = &y; Z*pz = &z;

void out(void) {
   px->f(); px->g();
   py->f(); py-> g();
   cout << '\n';
}

int main () {
      out(); px = py;
      out(); py = pz;
      out();  return 0;
}
```

6. В каких языках из перечисленных есть понятие «исключения»?

Приведите пример возникновения исключения в каком-либо из этих языков.

Си, Си++, Ада 83, Ада 95, Visual Basic, Оберон, Модула-2, C#, Delphi

7. Ниже приведена спецификация шаблонной функции перемножения матриц(двумерных массивов) на языке Си++. Напишите пример конкретизации этой функции, а также соответствующее описание родовой функции на языке Ада.

```
template <class T> Matrix<T>& MatMult (Matrix<T>& A,
Matrix<T>&B);
```

8. В каких областях памяти могут быть размещены объекты классов языка Си++?

2.5. Вариант 2007

1. В каких языках из перечисленных ниже при обработке исключительных ситуаций используется семантика возобновления? Объясните, что она означает.

Ада, Си, Си++, Оберон, Оберон-2, Модула-2, Java, C#

2. Что такое «явная реализация интерфейсов»? В каких языках она используется?

3. Дайте определение оператора цикла for в языке Java (все разновидности). Каким условиям должен удовлетворять класс-коллекция, чтобы его можно было использовать в этом операторе?

4. Дайте определение сопрограммы. Чем сопрограмма отличается от подпрограммы?

5. Что будет напечатано в результате работы следующей программы на Си++?

```
#include <iostream.h>
class X {
public:
   void g() { cout << 1 << ' '; }
   virtual void f() { g();}
};
class Y: public X {
public:
   virtual void g(){ cout<< 2<< ' ';}
   void f() { g(); }
};
class Z: public Y {
public:
   void g() { cout << 3 << ' '; }
   void f() { g(); }
};
X x; Y y; Z z; X*px = &x; Y*py = &y; Z*pz = &z;
void out(void) {
   px->f(); px->g();
   py->f(); py-> g();
   cout << '\n';
}
int main () {
      out(); px = py;
      out(); py = pz;
      out(); return 0;
}
```

6. В каких языках из перечисленных ниже отсутствует перечислимый тип данных?

Си, Паскаль, Ада 83, Ада 95, Си++, Оберон, Оберон-2, Модула-2

Опишите реализацию перечислимого типа данных в языке C#.

7. Опишите на языке Ада родовой модуль, реализующий абстрактный тип данных Queue (очередь). Реализацию процедур и функций писать не надо.

8. В каких из перечисленных ниже языков есть двусторонняя связь между модулями при раздельной трансляции? Объясните её смысл для какого-нибудь языка.

Ада, Си++, Оберон, Модула-2, Java, Delphi, C#

2.6. Вариант 2008 (пересдача)

1. Объясните смысл ключевого слова **sealed** в языке C#. В каких из перечисленных ниже языков есть соответствующее понятие?

Java, Ада 83, Ада 95, Си++, Delphi, Оберон, Оберон-2, Модула-2

2. Объясните, что означает термин «вложенные модули». В каких из перечисленных ниже языков есть соответствующее понятие?

Ада 83, Ада 95, Delphi, Оберон, Оберон-2

3. Напишите спецификацию абстрактного типа данных HashTable (перемешанная таблица, хэш-таблица) на языках Модула-2 и Java (тела методов и модуль реализации можно опустить).

4. Объясните, что означает термин «виртуальный метод»? В каких из перечисленных ниже языков есть соответствующее (или аналогичное) понятие?

Ада 83, Ада 95, Java, Delphi, Оберон, Оберон-2, Си, Си++, C#

5. Дайте определение абстрактного типа данных (АТД) и абстрактного класса (АК). Перечислите сходства и различия этих понятий. Приведите примеры АК и АТД на каких-нибудь языках программирования (только спецификации – тела процедур и функций писать не надо).

6. В каких из перечисленных ниже языков есть понятие «перегрузка имен» (или «перекрытие имен»)? Объясните, что оно означает (на примере какого-либо языка).

Чем перегрузка отличается от замещения?

Ада 83, Ада 95,Си, Си++, Оберон, Оберон-2, Модула-2, Java, Delphi, C#

7. Объясните, чем отличается понятие «структура» от понятия «класс» в языке C#?

8. Что означает ключевое слово **override** в языках C# и Delphi? Почему это ключевое слово (или аналогичное ему) отсутствует в языке Java?

III. ОТВЕТЫ, УКАЗАНИЯ И РЕШЕНИЯ

3.1. Вариант 2003

1.

```
B::f
B::g
B::h
A::h
B::f
B::g
B::h
B::h
-----
C::f
C::g
A::h
C::f
C::g
B::h
```

2. Ада 83, Ада 95, Delphi, C#

3. Конструкция языка Delphi «uses список_имен_модулей» служит для импорта всех имен, объявленных в интерфейсе модулей из списка. При этом импортированные имена становятся непосредственно видимыми (если нет конфликтов с именами из других модулей).

Конструкция языка Ада «use список_имен_пакетов» обеспечивает непосредственную видимость имен из спецификаций пакетов из списка (если нет конфликтов).

Сходство: конструкции обеспечивают непосредственную видимость имен из интерфейсов (спецификаций) при отсутствии конфликтов.

Различие: в Delphi «uses» импортирует имена из интерфейсов библиотечных модулей, в Аде импорт имен обеспечивается другими конструкциями, а «use» служит только для разрешения непосредственной видимости.

4. Си++.

Пример динамического массива в языке Java (или C#):

```
void f(int N) {
     byte [] dynArray = new byte [N];
     // ...обработка ...
}
```

Замечание: в языках Оберон и Модула-2 длина формальных параметров — открытых массивов является динамическим атрибутом. В других случаях длина массива — статический атрибут. В Аде формальные

параметры неограниченных типов-массивов также имеют динамический атрибут-длину (равно как и динамические массивы-локальные переменные).

5. Семантика возобновления: после обработки исключения управление может вернуться непосредственно в точку, где возникло исключение (варианты: на следующий оператор или на любой оператор из того же блока).

Пример языка: Visual Basic.

Семантика завершения: после возникновения исключения блок, в котором оно возникло, обязательно завершается. Обработка исключения происходит в блоках, вызвавших блок с исключением.

Пример языка: Си++.

6. В языке Ада 83 подпрограммы не могут быть параметрами подпрограмм (еще пример: Java).

7. Классы памяти:

- статическая;
- квазистатическая;
- динамическая.

В языке C# объекты классов размещаются только в динамической памяти.

8. Один из вариантов:

```
template <typename T, int size> class Stack
{
public:
     Stack() {top = 0;}
     void Push(T x);
     T Pop(T& x);
     bool IsEmpty();
     bool IsFull();
private:
     Stack (const Stack& s);
     T body[N];
     int top;
};
```

3.2. Вариант 2004

1.
X::f
X::g
X::g
Y::f
Y::g
Y::f

Y::f
X::g
Z::f
Y::g
Z::f

2. Ада 95, Си++, Оберон-2, Java, Delphi, C#.

3. Полностью эквивалентный фрагмент написать нельзя, поскольку Ада требует полной инкапсуляции структуры типа, а Оберон позволяет открывать поля структуры (в примере - I открыто, а J – закрыто). Однако можно на Аде написать написать подпрограммы доступа для I (get/set) и добиться того же эффекта.

```
type T is tagged private;
type T1 is new tagged T with private;
procedure P(X:T; L: integer);
procedure P(X:T1; L: integer);
```

Замечание: на языке Оберон процедуры P динамически привязаны к типу (T и T1 соответственно), однако на Аде динамическая привязка — это свойство не метода, а вызова. Поэтому разницы между динамически и статическими привязанными методами в Аде нет.

4. Ада, Delphi, C#
Язык C#:

```
try {
     // блок try
     ...
} finally {
     // блок  finally
     ...
}
```

Блок finally обязательно выполнится после завершения блока try независимо от того, нормально или аварийно (с возбуждением исключения) произойдет это завершение.

5. Класс S имеет два конструктора:

- явно описанный `explicit S(int);`
- сгенерированный конструктор копирования `S(const S&);`

Замечание: конструктор умолчания не генерируется, так как в классе есть явно описанный конструктор.

6. Конструкция package имеет вид:

package имя_пакета;

Она может быть только первой в единице компиляции. Смысл состоит в том, что класс, описанный в единице компиляции, относится к указанному пакету. Пакет является единицей дистрибуции Java-классов, а также единицей контекста.

7. Тип, имя которого экспортируется, но все поля — закрыты (не экспортируются).

```
TYPE OpaqueT* = RECORD I:TT; J: TTT END;
```

8.

```
void f()
{
     try {
          g(); h();
     } catch (E1){
          throw;
     } catch (E2){
          throw;
     } catch (E3){
          throw;
     } catch (...) {
          unexpected();
     }
}
```

3.3. Вариант 2005

1.

A::f
A::g
A::g
B::f
B::g

B::f

B::f
A::g
C::f
B::g
C::f

2. Семантика возобновления: после обработки исключения управление может вернуться непосредственно в точку, где возникло исключение (варианты: на следующий оператор или на любой оператор из того же блока). В языке Си++ реализована другая семантика: завершения, но в некоторых случаях семантика возобновления может быть смоделирована, например, в случае выделения возобновляемого ресурса (типа динамической памяти):

```
Resource GetResource() {
  for (;;)
    try {
      Resource r = … // попытка получить ресурс, например
                     // выделить память
      if (success) return r;
      throw NoResourceException();
    } catch (NoResourceException) {
      // попытка найти дополнительные ресурсы (например,
      // динамически собрать мусор)
      if (!success) throw;
    }
}
```

3. Язык Ада (реализация в виде двунаправленного списка):

```
generic
  type T is private;
package G_Deque is
  type Deque is limited private;
  procedure PushRight(Deq: inout Deque; X:T);
  procedure PushLeft(Deq: inout Deque; X:T);
  procedure PopRight(Deq: inout Deque; X: out T);
  procedure PopLeft(Deq: inout Deque; X: out T);
  procedure Init(Deq: out Deque);
  procedure Destroy(Deq: inout Deque);
  function IsFull(Deq: Deque);
  function IsEmpty(Deq: Deque);
  -- другие процедуры ...
private
  type PLink is access;
  type Link is record inf : T; next, prev : PLink; end record;
  type PLink is access Link;
  type Deque is record Left, Right: PLink; end record;
end G_Deque;
```

Язык Java;

```
interface IDeque<T>
{
     void PushLeft(T x);
     void PushRight(T x);
     T PopLeft();
     T PopRight();
     bool IsFull();
     bool IsEmpty();
     // другие функции
}
```

Замечания:

а). Обобщенные конструкции употреблять не обязательно (надо только написать, что тип T должен быть непосредственно видимым в точке описания типа Deque). Хотя обобщения здесь подходят больше.

б). Структуру типа в Аде полностью выписывать необязательно. Главное — указать наличие приватной части, например:

```
private
  ...
  type Deque is …; -- структура типа Deque
end G_Deque;
```

в). Для языка Java можно выписать не инетерфейс, а конкретный класс с приватной структурой и публичными функциями-операциями. Тела функций в этом случае можно не выписывать. Но интерфейс в данном случае больше подходит к понятию абстрактного типа данных.

4. Языки;

Ада 95, Си++, Java, Delphi, C#

Например, в языке Си++ абстрактная функция — это чистая виртуальная функция. Она не обязана иметь тела и должна быть обязательно замещена в каком-либо производном классе.

5. Ключевое слово super означает в языке Java ссылку на базовый класс. Такое же понятие есть в языке C#, но оно называется base.

Пример на Java:

```
class A
{
     public A(int I) { … }
     ...
}

class B extends A
{
     public B() { super(0); … }
     ...
}
```

Пример на C#:

```
class A
{
     public A(int I) { … }
     ...
}

class B : A
{
     public B() : base(0) { … }
     ...
}
```

6. Delphi, C#

«Свойство» - это член класса, который с точки зрения обращения к нему выглядит как член-данное, но с точки зрения реализации представлен двумя методами, один из которых возвращает значение свойства, а второй — устанавливает его значение. При этом один из методов может отсутствовать, делая недоступной соответствующую операцию над свойством.

Пример для языка Delphi – целое свойство Prop:

```
type PropSample = class
  …
  private
     procedure SetPropVal(V : integer);
     function GetPropVal:integer;
  public
     property Prop: integer read SetPropVal write GetPropVal;
  …
 end;
```

7. Ада 83, Ада 95, Си++, Java, Delphi, C#

Понятие «перегрузка» означает, что одному имени в одной области видимости может соответствовать несколько определений. В современных языках программирования перегружаться могут только имена подпрограмм, но не типов, переменных, модулей.

8. Основное отличие деструктора языка C# от деструктора Си++ состоит в том, что неизвестен момент его вызова. Деструктор (в действительности — финализатор) вызывается сборщиком мусора в момент утилизации памяти, занимаемой объектом. Для ряда объектов деструктор вообще может быть не вызван. В Си++ момент вызова деструктора достаточно точно определен.

3.4. Вариант 2006

1. Ада, Модула-2, Паскаль

Объединение типов (или запись с вариантами) — это конструкция, объединяющая в один тип несколько различных структур (вариантов). Все варианты в объединении начинаются с одного адрес и занимают одну и ту же память.

Размеченное объединение типов содержит одно выделенное поле (дискретного типа данных) — общее для всех вариантов. Такое поле называется дискриминантом. Значение дискриминанта определяет определяет, по какому варианту выделена память в переменной-экземпляре размеченного объединения.

2. Указатели языка Ада 83 ссылаются только на объекты из динамической памяти. Указатель языка Си++ может ссылаться на любой объект данных (динамический, локальный, статический), что может приводить к труднообнаружимым ошибкам.

Также в языке Ада отсутствует адресная арифметика (арифметические операции над указателями), что также уменьшает вероятность появления ошибки работы с памятью.

3. Си, Си++

Раздельная трансляция означает то, что программа разбивается на части — физические модули или единицы компиляции. Каждая единица может или обязана транслироваться отдельно от остальных.

Независимая раздельная трансляция означает то, что транслятор не обладает информацией об уже оттранслированных единицах и поэтому не может проверить корректность межмодульных связей.

4. Конструкция where используется в языке C# для определения ограничений на параметры родовых (другое название - обобщенных) конструкций.

Ее вид: where имя_типа : список_ограничений.

Виды ограничений:

- интерфейс — означает, что параметр-тип должен реализовывать этот интерфейс;
- имя класса (может быть только одно такое ограничение в списке) — означает, что параметр-тип должен быть наследником этого класса;
- struct или class – означает, что параметр-тип должен быть структурой или классом;
- new() - означает, что параметр-тип должен иметь конструктор умолчания (без параметров).

5.
1
1
2
1
2
3 3
1 1 2 2
2 2 2 2
2 2 3 3

6. Си++, Ада 83, Ада 95, Visual Basic, C#, Delphi
Пример для языка Delphi:

```
if ptr = nil then
     raise Exception.Create('Invalid pointer');
```

7. Конкретизация на языке Си++:

```
Matrix<float> b,c;
…
Matrix<float> a = MatMult(b,c);
```

Язык Ада:

```
generic
  type T is private;
  with function "+"(x,y:T) return T (<>);
  with function "*"(x,y:T) return T (<>);
  type Matrix is private;
function G_MatMult(A,B: Matrix) return Matrix;
```

8. Объекты классов языка Си++ могут быть размещены в статической, квазистатической, динамической памяти.

3.5. Вариант 2007

1. Семантика возобновления при обработке исключений состоит в том, что после обработки исключения управление может вернуться непосредственно в точку, где возникло исключение (варианты: на следующий оператор или на любой оператор из того же блока, где возникло исключение).

В перечисленных в условии языках используется другая семантика: завершения, либо вообще отсутствует понятие реакции на исключение.

2. Явная реализация интерфейса означает, что вызов метода интерфейса может происходить только через ссылку на интерфейс, но не

может происходить через ссылку на класс, реализующий интерфейс. Перед вызовом интерфейсного метода необходимо явно преобразовать ссылку на объект реализующего класса к ссылке на интерфейс. Концепция явной реализации полезна, например, при конфликте имен между унаследованными интерфейсами. Используется, например, в C#.

```
interface ISomeInterface
{
  void F();
}
class CoClass: ISomeInterface
{
  ISomeInterface.F() {
     System.Console.WriteLine("Явно реализованный метод");
  }
  ...
}
  ...
CoClass c = new CoClass();
c.F(); // ошибка: нельзя вызывать явно реализованный метод
       // интерфейса через ссылку на объект
(ISomeInterface)c.F(); // все нормально
```

3. В языке Java используется 2 формы оператора цикла for.

Первая форма полностью соответствует оператору for языка Си++:

```
for (e1;e2;e3) S
```

Вторая форма появилась в 2005 году и используется для поэлементного просмотра коллекций (цикл for-each). Она имеет вид;

```
for (T v:Coll)S
```

Здесь Coll — коллекция элементов (типа T или приводимых к типу T). Переменная v на каждой итерации цикла принимает значение очередного элемента коллекции.

Для того, чтобы объекты класса-коллекции могли появляться в цикле for-each, класс должен реализовать интерфейс Iterable.

4. При вызове сопрограммы управление передается в точку, непосредственно следующую за местом, где оно покинуло сопрограмму. В подпрограммах управление всегда начинается с первого оператора, и это не зависит от того, в какой точке управление покинуло подпрограмму в прошлый раз.

5. 1 1 2 2
   2 1 2 2
   2 1 3 3

6. Оберон, Оберон-2

Перечислимый тип в C# имеет вид:

```
enum T : базовый_целый_тип {
     список_констант
}
или
enum T {
     список_констант
}
```

По умолчанию базовый_целый_тип — это int.

Каждая константа в списке может быть инициализирована своим значением (как в Си++). К перечислениям нельзя применять арифметические и побитовые операции, но если перед перечислением стоит атрибут [Flags], то к элементам перечисления применимы побитовые операции | и &.

Константы из перечислимого типа видимы только потенциально и при обращении должны уточняться именем типа: тип.имя_константы.

Пример:

```
enum Color : long
{
  Red,
  Green = 50,
  Blue
}

Color c = Color.Red;
```

7.

```
generic
  type T is private;
  Size : integer;
package G_Queue is
  type Queue is limited private;
  procedure Enqueue(Q: inout Queue; X:T);
  procedure Dequeue(Q: inout Queue; X:T);
  procedure Init(Q: out Queue);
  procedure Destroy(Q: inout Queue);
  function IsFull(Q: Queue);
  function IsEmpty(Q: Queue);
  -- другие процедуры ...
private
  type Queue is record
     Left, Right: integer;
     body : array(1..Size) of T;
  end record;
end G_Queue;
```

8. Ада

При односторонней связи (импорт-экспорт) модуль, экспортирующий имена, не зависит от импортирующих (клиентских) модулей. При двусторонней связи оба модуля зависят друг от друга. В языке Ада двусторонняя связь используется при раздельной трансляции вложенных модулей.

Вложенный модуль обозначается «заглушкой» во внешнем модуле:

```
procedure Outer is
  -- заглушка
  procedure Inner is separate;
  . . .
end Outer;
```

При трансляции вложенный модуль снабжается заголовком, связывающим его с объемлющим модулем:

```
separate(Outer)
procedure Inner is
 . . .
end Inner;
```

Связь «заглушка-заголовок» - пример двусторонней связи.

3.6. Вариант 2008

1. Ключевое слово sealed может стоять перед виртуальным методом или классом. В первом случае оно означает, что метод нельзя замещать в производных классах, во втором — что класс нельзя наследовать. Из перечисленных в условии языков аналогичное понятие есть в языке Java (final).

2. Вложенность модулей означает, что определение одного модуля (внутреннего) находится внутри другого (внешнего). Из перечисленных в условии языков вложенные библиотечные модули могут быть только в языке Ада (как 83, так и 95). В Обероне и Delphi вложенными могут быть только подпрограммы (которые не являются библиотечными модулями).

3. Язык Модула-2.

```
DEFINITION MODULE HashTables;
  FROM Types IMPORT KeyType, ElementType;
  TYPE  HashTable;
  PROCEDURE Init(VAR T:HashTable);
  PROCEDURE Destroy(VAR T:HashTable);
```

```
  PROCEDURE Lookup(VAR T:HashTable;
                   Key: KeyType; VAR X:ElementType):BOOLEAN;
  PROCEDURE Add(VAR T:HashTable; Key: KeyType; X:ElementType);
  PROCDURE Remove(VAR T:HashTable; Key: KeyType):BOOLEAN;
  VAR Done: BOOLEAN;
END  HashTables.
```

Язык Java:

```
interface IHashTable : Iterable
{
     ElementType Lookup(KeyType Key);
     void Add(KeyType Key, ElementType El);
     bool Remove(KeyType Key);
}
```

4. Ада 95, Java, Delphi, Оберон-2, Си++, C#

Виртуальность метода означает динамическое связывание метода при вызове метода через ссылку на объект (базового) класса. Будет вызван метод для объекта, на который в настоящее время указывает ссылка. Этот объект может относиться не к базовому, а к производному классу. если в производном классе метод замещен, то будет вызван не метод из базового класса (как в случае невиртуальных методов), а его заместитель .

5. Абстрактный тип данных (АТД) — это тип с полностью инкапсулированной структурой. Использовать объекты АТД возможно только при помощи явно определенных в интерфейсе типа операций.

Абстрактный класс (АК) — это класс, содержащий хотя бы один абстрактный метод.

Прямой связи между АК и АТД нет. АТД может быть абстрактным классом, а может и не быть. Аналогично, АК может иметь инкапсулированную структуру, а может и не иметь.

Пример АТД можно найти в ответе на задачу 3 этого варианта. Пример АК (язык Java):

```
     interface IDrawable
     {
       void Paint();
     }

     abstract class UIControl : IDrawable
     {
          ... // нет реализации метода  Paint()
     }

     class EditControl:  UIControl
     {
       public void Paint() { ... // реализация метода Paint()}
       . . .
     }
```

6. Ада 83, Ада 95,Си++, Java, Delphi, C#

Понятие «перегрузка» означает, что одному имени в одной области видимости может соответствовать несколько определений. В современных языках программирования перегружаться могут только имена подпрограмм, но не типов, переменных, модулей. Пример на языке Си++:

```
class X {
public:
  void f();
  void f (int)
};
X a;
a.f();  // первая функция
a.f(0); // вторая функция
```

Отличие перегрузки от замещения состоит во-первых, в том, что перегрузка обрабатывается статически (на этапе трансляции), в во-вторых, при замещении речь идет о разных областях видимости: базовый класс с объявлением виртуального метода (объемлющая область видимости) и производный класс с замещающим методом (вложенная область видимости).

7. Структура в языке C# является типом-значением. Имена структур не являются ссылками (как имена объектов-классов), а обозначают непосредственно объект.

Структуры не могут наследоваться, не могут быть наследовать сами (только по умолчанию от класса Object). Также структуры не могут иметь явный конструктор умолчания.

8. Ключевое слово override означает, что метод, в объявлении которого оно появляется, является заместителем виртуального метода из базового класса.

В языке Java такая конструкция отсутствует, поскольку все методы имеют динамическое связывание, поэтому любой метод, имя и сигнатура которого совпадают с методом их базового класса, является его заместителем.

Список литературы

Основная литература

1. Кауфман В.Ш. Языки программирования: концепции и принципы. – М.: Радио и связь, 1993.
2. Бен-Ари М. Языки программирования. Практический сравнительный анализ. – М.: Мир, 2000.
3. Пратт Т., Зелковиц М. Языки программирования: разработка и реализация. – СПб.: Питер, 2002.
4. Себеста Р.У. Основные концепции языков программирования. – М.: Издательский дом «Вильямс», 2001.

Дополнительная литература

1. Страуструп Б. Дизайн и эволюция Си++. – М.: ДМК Пресс, 2000.
2. Страуструп Б. Язык программирования Си++. – М., СПб. : «Издательство БИНОМ» - «Невский Диалект», 2001.
3. Вандевурд Д., Джосаттис Н.М. Шаблоны Си++, справочник разработчика. - М.: Издательский дом «Вильямс», 2003.
4. Арнолд К., Гослинг Д. Холмс Д. Язык программирования Java. - М.: Издательский дом «Вильямс», 2001.
5. Вирт Н. Программирование на языке Модула-2. – М.: Мир, 1987.
6. Джехани Н. Язык Ада. – М.: Мир, 1988.
7. Шилдт Г. C# 2.0. Полное руководство. – М.: ЭКОМ Паблишерс, 2007.

Содержание