Лекция 4-5. Вычислительный контекст, состояния

Roadmap – где мы сейчас находимся и куда движемся

(Заметка на полях к обсуждению про систему типов C++) From c++ standard 4.3/1. Floating-Integral conversion

A prvalue of a floating point type can be converted to a prvalue of an integer type. The conversion truncates; that is, the fractional part is discarded. The behavior is undefined if the truncated value cannot be represented in the destination type

модульность – принцип организации программ при котором она может быть разбита на составные части так, что каждая из этих частей может разрабабываься и отлаживаться независимо от других

Оператор присваивания – позволяет изменять значение связанное с именем.

Модель вычислений

подстановочная (substitution)	с окружениями (environmental)
``(define (factorial n) (cond ((= n 1) 1) (else (* n (factorial (- n 1)))))) // или factorial(x [x>=1]) { if(x==1) return 1; else return x* factorial(x-1) }```	```factorial(x [x>=1]) { result = 1; while(x>1) { result *=x; x--; } return result; }```
В функциональных языках (без присваивания)	**Императивные (с присваиванием)**
Из чего состоит программа: • функции первого класса • рекурсия • условные выражения Особенности: • результат вычисления функции завист только от ее аргументов • нет побочных эффектов • многократный вызов функции с теми же аргументами дает тот же самый результат	Из чего состоит программа: • переменные и присваивания • условные выражения • Циклы Особенности: • есть побочные эффекты • большинство команд (инструкций) переводят программу из одного состояния в другое • значения возвращаемые функциями, зависят от истории их вызова.

Полагается на то, что имена ссылаются на значения объектов -> значение объекта не может быть изменено	Полагается на то, что имена ссылаются на место хранения объектов -> объект по заданному месту хранения может быть перезаписан

Откуда берутся состояния и время

- одна из задач программирования – управление сложностью
- соблюдение принципа модульности позволяет управлять сложностью
- вычислительные программы (моделирующие математические функции и правила) не одинственные – есть необходимость создавать программы моделирующие физические объекты и процессы
- подход к моделированию физических объектов:
  - структура программы соответствует структуре моделируемой системы
  - для каждого объекта строится соответсвующий ему вычислительный объект
  - для каждого действия выдлеляется символьная операция (функция или процедура)
- НО:
  - подстановочная модель перестает работать, так как поведение объектов зависит от времени
  - состояние объектов меняется от времени или от взаимодействия с другими объектами

Контрольные вопросы:

- какая модель вычислений соответствует функциональным возможностям языка программирования
- какая модель вычислений используется при императивном стиле программирования
- Каков смысл переменной при функциональном стиле написания программ
- -- при императивном
- Чем рекурсия отличается от цикла?

Пример: снятие денег со счета

```
withdraw(amount) {
 static balance = 100 // используем static для обозначения тог
                      // что balance это хранимое состояние объекта (этой функции)
 if(balance >= amount) {
  balance -= amount;
  return amount;
 }
 print "Not enough money"
}
// многократный вызов функции с теми же аргументами будет всегда давать различные
значения
// здесь balance -- определяет внутреннее состояние
```

Присваивание – влечет сложности с определеним корректности программ, однако обеспечивает водможность модульности. Это размен.

Смысл тождественности объектов

withdraw1(x) { return 100-x }	withdraw2(x) { return 100-x }	Одно и то же? withdraw1(25) withdraw1(25) withdraw2(25)
withdraw3(x) { static balance = 100; return balance-x; }	withdraw4(x) { static balance = 100; return balance-x; }	withdraw3(25) withdraw3(25) withdraw4(25)

Язык программирования является референциально прозрачным (reverentially transparent), если в любом выражении замена объекта на такой-же (идентичный? см. обсуждение ниже) не влияет на результаты вычислений.

Обсуждение:

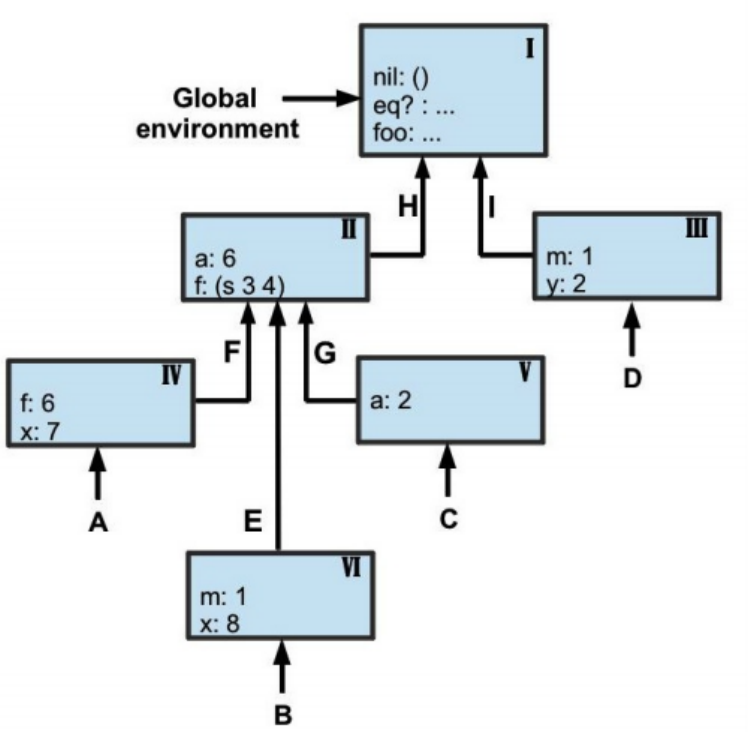
- что такое идентичный объект?
- как в реальном мире проверить/доказать – являются ли 2 объекта одним и тем же?
  - изменить один и посмотреть не изменился ли другой

Порядок присваивания повышает сложность – результат зависит от порядка.

```
factorial(x [x>=1]) {
 // result = 1;       //!
 while(x>1) {
  result *=x;
  x--;
 }
 result = 1;          //!
 return result;
}
```

Окружение:

- связывание переменных – сопоставление имени переменной ее значению
- таблица связываний – набор связок между именами и значениями
- кадр – таблица (возможно пустая) связываний
- окружение – последовательность кадров
  - каждый кадр имеет не более одного связывания для каждого имени (переменной)
  - кадр содержит ссылку на объемлющее (внешнее окружение)
  - считается, что всегда существует самое внешнее, глобальное окружение
  - значение переменной по отношению к данному окружению – это значение связывания, которое находитмя в первом кадре окружения, содержащего такое связывание (движение изнутри наружу)
  - окружение определяет контектст внутри которого вычиляются выражения. Можно конечно сказать, что операция 1+3 может быть вычислена без контекста ... (однако кто знает что такое значит символ `+`?)
  - глобальное окружение содержит все предопределенные связывания



- переменные, а следовательно и выражения не имеют значения само по себе
- если не один из кадров не имеет связывания для переменной, то говорят, что переменная несвязана
- если существует более одного фрейма, где переменная связана, то считается что действует ближайшее связывание, а остальные связывания затенены (shadowed, перекрыты)
- Операции с именами:
  - define - создает связку со значеним в текущем фрейме
  - := осуществляет поиск связывания и если находит, то меняет значение, если нет – ошибка.

Вычисление (выполнение) процедуры с аргументами

- создание нового фрейма, в котором аргументы (имена) связываются с их реальными значениями
- формируется новое окружение, начинающееся с вновь созданного фрейма (созданный фрейм указывает на предыдущее окружение)
- тело процедуры выполняется в новом окружении

Контрольные вопросы

- каким свойством обладает язык программирования, если замена объекта на идентичный не приводит к изменению результатов вычислений
- что такое побочный эффект?
- что такое кадр/фрейм
- что такое окружение
- может ли имя/переменная быть связанно более одного раза в заданном окружении (можно ли составить такое окружение в котором переменная связана более одного раза)?
- может ли переменная быть связанной более одного раза в одном фрейме?
- что такое контекст?

References

- Абельсон Сассман, SICP гл.3