Лекция 2-3. Исполение программ. Система типов.

Исполнение

- компиляция
- интерпретация

Контрольные вопросы:

- Можно ли скомпилировать интерпретатор? компилятор?
- Перечислите 2 интерпретируемых языка
- Может ли интерпретируемая программа быть выполнена без интерпретатора
- Может ли скомпилированная программа быть выполнена без компилятора
- Можно ли скомпилировать программу на интерпретируемом языке

Система типов

def: множество правил и соглашений, в соответствии с которыми каждому элементу/объекту программы (функциям, переменным, модулям, выражениям, классам,...) присваивается соответствующий атрибут именуемый типом.

зачем нужна система типов? Для того чтобы проверять корректность согласованности частей программы, аналогично тому как в физике проверятся размерность выражения. Если система типов определена, то проверку можно выполнять автоматически.

Типизация: процесс проверки ограничений устанивливаемых системой типов, другими словами проверка согласования типов.

NB: есть сложные и простые типы

- простые (не предоставляющие программисту доступа к внутренней структуре)
  - string - в некоторых языках
  - доступ к внутренней структруе все же есть, но опосредованно через функции. Сравни со структурами
  - примеры:
    - всевозможные числа:int, double, float, boolean,
    - символы
    - ссылки, указатели
  - сложные составные
    - массивы
    - структуры
    - классы
    - комплексные числа
    - замыкания, продолжения в функциональных языках

Объекты первого класса в языке программирования (first class object)

def:

- Объект первого класса (first class object):
  - может быть присвоен, сохранен в структуре данных
  - быть переданы как парамеры
  - возвращен из функции
  - создан во время исполнения программы
  - независим от именования

Типизация:

- статическая – на этапе компиляции
- динамическая – в процессе работы
  - утиная типизация vs типизация наследованием
- сильная (без потери значимости)
- слабая (с возможностями потери значимости)

Приведение типов

- Приведение типов
  - с потерями данных (в слабо типизированных языках)
  - без потерь данных (в сильно типизированнных языках)
  - Виды:
    - явное
    - неявное
    - заданы функции приведения

Вывод типов (type inference) – способность компилятора на основе системы типов определить тип выражения.

Типизированный язык – в котором запрещено выполнять операции над несовместимыми типами

Нетипизированный (безтиповый) язык – в котором разрешено выполнять операции над несовместимыми типами (ассемблеры, Forth)

Контрольные вопросы:

- приведите пример 2 простых типов языка python?
- как называется типизация выполняемая на этапе компиляции?
- может ли в языке программирования присутсвовать как статическая так и динамическая типизация?
- встречаются ли нетипизированные языки?
- может ли в интерпретируемом языке использоваться статическая типизация?