МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

САНКТ**-**ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**,** МЕХАНИКИ И ОПТИКИ

Кафедра Информатики и прикладной математики

Лабораторная работа №6 (8 вариант)

По предмету «Верификация моделей программ»

Выполнил:

студент гр. № Р4115

Назукин Д. Е.

Проверил :

Кореньков Ю. Д.

Санкт-Петербург

2017

**Цели**

Освоение верификации программ с учетом полиморфизма диспетчеризации.

**Задачи**

Добавить в разработанный за предыдущие задания программный комплекс поддержку перегрузки функций/методов и поддержку полиморфизма подтипов с переопределением реализаций логики (опционально виртуальные методы).

**Описание работы**

Под перегрузкой функции понимается, определение нескольких функций (две или больше) с одинаковым именем, но различными параметрами. Наборы параметров перегруженных функций могут отличаться порядком следования, количеством, типом. Таким образом перегрузка функций нужна для того, чтобы избежать дублирования имён функций, выполняющих сходные действия, но с различной программной логикой.

В объектно-ориентированном программировании полиморфизм подтипов (или полиморфизм включения) представляет собой концепцию в теории типов, предполагающую использование единого имени (идентификатора) при обращении к объектам нескольких разных классов, при условии, что все они являются подклассами одного общего надкласса (суперкласса). Полиморфизм подтипов состоит в том, что несколько типов формируют подмножество другого типа (их базового класса) и потому могут использоваться через общий интерфейс.

**Аспекты реализации**

Для реализации перегрузки функций на этапе компиляции фиксируется только название вызываемой функции, а конкретная функция уже выбирается во время выполнения кода по типу аргументов и их количеству. Это сделано так, потому что при динамической типизации на этапе компиляции мы не знаем типы аргументов, которые передадутся в функцию, и соответственно не знаем, какая из перегруженных функций должна быть вызвана. Также типы аргументов в сигнатуре функции фиксируют типы аргументов, передаваемых в эту функцию, т.е. в сигнатуре функции типы аргументов «long», то передать в эту функцию можно только «long» значения.

Было реализовано наследование с возможностью переопределения логики функций. Для возможности вызова методов базового класса в мета-данные класса добавлена ссылка на базовый класс. Поиск вызываемого метода происходит рекурсивно (изначально в классе объекта, потом если есть базовый класс, в нем и т.д.)

**Результаты**

Вход программы файл с функциями (приведены только основные функции):

public function sum(A as long, B as long, f)  
 sum = A+B;  
 print("long sum= " + sum + "\n", f);  
end function  
  
public function sum(A as string, B as string, f)  
 sum = A+B;  
 print("string sum= " + sum + "\n", f);  
end function  
  
public function sum(A as bool, B as bool, f)  
 sum = A||B;  
 print("bool sum= " + sum + "\n", f);  
end function

class User  
 private id, name  
  
 public function toString()  
 toString = "User - " + "id: " + getId() + " name: " + name;  
 end function  
  
 public function getName()  
 getName = name;  
 end function  
  
 public function setName(\_name as string)  
 name = \_name;  
 end function  
  
 public function getId()  
 getId = id;  
 end function  
  
 public function setId(newId)  
 id = newId;  
 end function  
  
 public function New(\_id as long, \_name as string)  
 id = \_id;  
 name = \_name;  
 end function

end class

class MegaUser inherits User  
 public function New(\_id as long, \_name as string)  
 setId(\_id);  
 setName(\_name);  
 end function  
 public function toString()  
 toString = "MegaUser - " + "id: " + getId() + " name: " + getName();  
 end function  
end class

function main()  
 f = CreateFile("out.txt", 0x40000000, 0, 0, 2, 0, 0);  
  
 a.sum("55","44",f);  
  
 a.sum(55,44,f);  
  
 a.sum(true,false,f);

b = new User(3, "Ivan");  
 print(b.toString() + "\n", f);

b = new MegaUser(5, "Alex");  
 print(b.toString() + "\n", f);

main=0;  
end function

В результате будет создан файл out.txt и в него записан результат:

string sum= 5544  
long sum= 99  
bool sum= true

User - id: 3 name: Ivan

MegaUser - id: 5 name: Alex

**Вывод**

В ходе работы реализована добавлена в разработанный за предыдущие задания программный комплекс поддержка перегрузки функций и поддержка полиморфизма подтипов с переопределением реализаций логики