PA3 ++

Practice.

COOL-TREE.HANDCODE.H

// Define lists for environment

Класс для списка всех классов программы

+ typedef SymbolTable<Symbol, Class__class> class_list_type;

Класс для списка аттрибутов класса

+ typedef SymbolTable<Symbol, tree_node> attr_list_type;

Класс для списка методов одного класса

+ typedef SymbolTable<Symbol, tree_node> method_list_type;

SEMANT.H

```
class ClassTable {
…
+        class_list_type *class_list;
```

То есть в таблице классов теперь будет храниться их список

COOL-TREE.H

Заполнена структура дерева.

В основном добавленны поля type, геттеры get_type и вызовы semant.

Semant принимает 3 параметра:

- class_list_type* - список всех классов
- attr_list_type* - список аттребутов текущего класса
- method_list_type* - список методов текущего класса

COOL-TREE.H особенности

Немного отличаются дополнения для следующих классов:

- **branch_class** : метод для получения типа выражения - get_expr()
- **formal_class** && **attr_class** : метод для получения имени get_name()
- **method_class** :
  - get_formals() – получение списка параметров
  - get_name() – получение имени
- **class__class** :
  - Дополнительные переменные: attr_list method_list
  - get_name()
  - get_parent()
  - fill_table(class_list)
  - Методы для взятия аттрибутов и методов: get_attr() && get_methods()

TREE.H hack

```
  class tree_node {
…
+       virtual Symbol get_type();
+       virtual Symbol get_name();
```

Сделанно для того, чтобы гарантировать возможность возвращения типов из всех наследников tree_node.
Можно было сделать их чисто виртуальными (=0 - нет, это не смайлик)

SEMANT.CC welcome ! STL

DUMP – дефайн для отладочной печати – в случае падения возможно отдебажить принтами.

Добавление библиотек (ассерты – лишние, map, set – ассоциативные контейнеры) ***map*** – используется для поиска LUB, ***set*** – для временного хранения списков в семантическом анализе методов и case.

В самом начале реализованны методы

- LUB – поиск самого близкого родителя
- find_attr – поиск аттрибута, в том числе и у родителей.

STL – два слова.

За что любят C++ ?

Standart template library – стандартная библиотека шаблонов.

Т.к. шаблонов – то есть возможность кастомизировать + header-only
Часто используемые:
Контейнеры – string, vector, array, map, list, queue, e.t.c
Библиотеки работы с файлами и вводом-выводом – iostream, file
Библиотеки работы с потоками – thread, mutex, atomic
Алгоритмы и математические функции – math, algorithm
Прочее – regex, random, e.t.c

LUB

```
Symbol lub(class_list_type *class_list, Symbol A, Symbol B) {
   std::map<Symbol, Class__class*> a_list;
   Class__class* a = class_list->lookup(A);
   for (; a != NULL;) {                                  // Здесь заполняется список родителей A
      a_list[a->get_name()] = a;
      if (a->get_name() == Object)
         break;
      a = class_list->lookup(a->get_parent());
   }
   Class__class* b = class_list->lookup(B);
   for (; b != NULL && !a_list.empty();) {               // Здесь ищется первое общее совпадение
      if (a_list.find(b->get_name()) != a_list.end()) {  // среди заполненного списка и родителей B
         return b->get_name();
      }
      b = class_list->lookup(b->get_parent());
   }
   return Object;
}
```

find_method && find_attr

```
tree_node* find_method(
            class_list_type *class_list,// Список всех классов
            Symbol A,                   // Имя класса
            Symbol method) {                    // Имя метода
   Class__class* a = class_list->lookup(A);
   method_list_type* list = a->get_methods();
   for (;;) {
      tree_node* node = list->lookup(method);
      if (node != NULL) {
         return node;                   // Нашли
      }
      if (a->get_name() == Object)
         break;                 // Не нашли
      a = class_list->lookup(a->get_parent()); // Берем родителя
      list = a->get_methods();
   }
   return NULL;
```


TO DIFF