Can pro13/02/2014 2 hr. Modificadas las Invalid Expression Exception y agregada aclaración que se debe usar Typedef en las pruebas.” ->” será tratado como “.” . Ahora puede leer varios atributos declarados en la misma línea de forma correcta.

14/02/2014 3hr. Ya guarda los atributos globales (solo numéricos,no soporta otro tipo) ,pero todavía no los incluye en las heurísticas ni los muestra. No corre todavía con Aeropuerto.c . Ahora lee números de mas de 1 cifra. Ahora puede leer caracteres encerrados entre apostrofes. Ahora soporta funciones que devuelven estructuras.

15/02/2014 2hr . Muestra atributos globales y soporta variables con “\_” . Funciones en el main como system(“cls”) deberían ser evitadas ya que no las puede leer bien todavía, supongo por el echo de que sea una función q en realidad no esta registrada ya que es de uno de los modulos y por lo tanto no la reconoce. Duda con q que se quiere mostrar de la main. Ya carga el tamaño de los arreglos bien para los adt,pero no para las candidate class.

18/02/2014 5hrs. Muestra y carga los modulos de los modulos y sus funciones. Muestra el tamaño de los arreglos correctamente y no muestra mas arreglos que no son.

19/02/2014 3hrs. Agregado explorador de archivos. Arreglados varios bugs. Imprime el output con el nombre del archivo. Probado University. Duda con MainFuction, que es cada cosa, que se quería hacer ,etc.

20/02/2014 2hrs. Carga la mainFunction como una función normal. Implementa todas las heurísticas. Duda con la heurística 5 ¿ Es necesaria? Tomo las variables globales como las que están fuera de las funciones, no me parece necesario tomar también las del main. De ser necesario, es cuestión de agregar unas líneas y listo.

26/02/2014 2hrs. Ya puede procesar multiplicaciones. Falta restas y demás ,al menos en las constantes globales. GlobalConstants with names that includes previous GlobalConstants as "TAM","TAM2" shouldn't be used since parser recognizes TAM, replaces for its value and then the 2.

27/02/2014 1hr. Correcion de bugs. Evitar definir mas de 1 vez una constant en todo el projecto, ya que el parser la reemplaza. Entrega output de mixadt(1),makewav y university.

18/03/2014 1hr. Ahora puede procesar funciones con arrays como argumentos.

//comienzo 2da etapa

10/04/2014 1hr. Formateo de lotes de prueba para coincidir con el formato aceptado por el parser.

12/04/2014 1 hr Formateo de lotes de prueba para coincidir con el formato aceptado por el parser. Ya acepta estructuras autoreferenciables, siempre y cuando sean en el formato “typedef struct{}nombre;

13/04/2014 1hr. Modificados , formateados y parseados los 10 test cases. Solucionados algunos bugs.

20/04/2014 1:30hr. Se llego a cubrir : \* En el parser desagregar la información de la columna modules. Y poner: “standard modules” y “modules”.

\* Mejorar el trabajo de las variables globales:

o Las variables que están dentro de un módulo NO SERÁN VARIABLES GLOBALES. Es decir, no pueden estar en la lista general de variables. Sino que deben ser “atributos” de la clase que representa al módulo.

o En la lista general de variables deberán estar:

\* Las variables que están definidas fuera del main(), en el archivo que contiene el main().

\* Aplicar la misma lógica a los #define.

\* Que el parser sea capaz de leer todos los módulos (includes) sin un orden estricto. Ahora, el listado de includes tiene que ser una sola manera. Para entender esto, ver el problema que se genera en los casos 2 y 4. Si se le altera el orden a las instrucciones 4 a 7 del caso 2, el parser no funciona. Ahora si se ordena para que compile, el parser no funciona. Y viceversa.

25/4/2014 2hr . Se imprimen las variables definidas adentro del main. Se borran los comentarios y se ignoran las modificaciones de atributos de adts en el main. Test Case MovieClub formateado. Crea CCD a partir de fprintf(falta testear y agregar fwritef). Crea CCD a partir de modulos.